대 한 민 국 특 허 청
KOREAN INTELLECTUAL PROPERTY OFFICE

별첨 사본은 아래 출원의 원본과 동일함을 증명함.

This is to certify that the following application annexed hereto is a true copy from the records of the Korean Intellectual Property Office.

출 원 번 호 : 10-2003-0063342
Application Number

출 원 년 월 일 : 2003년 09월 09일
Date of Application SEP 09, 2003

출 원 인 : 삼성전자주식회사
Applicant(s) SAMSUNG ELECTRONICS CO., LTD.

2004 년 03 월 25 일

특 허 청
COMMISSIONER

【서지사항】

【서류명】 특허출원서
【권리구분】 특허
【수신처】 특허청장
【참조번호】 0001
【제출일자】 2003.09.09
【발명의 명칭】 플래시 메모리의 오류블록 관리방법 및 장치
【발명의 영문명칭】 METHOD AND APPARATUS FOR MANAGING BAD BLOCK IN FLASH MEMORY
【출원인】
【명칭】 삼성전자 주식회사
【출원인코드】 1-1998-104271-3
【대리인】
【성명】 김동진
【대리인코드】 9-1999-000041-4
【포괄위임등록번호】 2002-007585-8
【발명자】
【성명의 국문표기】 윤송호
【성명의 영문표기】 YOON,Song Ho
【주민등록번호】 741030-1245819
【우편번호】 420-031
【주소】 경기도 부천시 원미구 상1동 반달마을 극동신라아파트 1843-707
【국적】 KR
【발명자】
【성명의 국문표기】 김장환
【성명의 영문표기】 KIM,Jang Hwan
【주민등록번호】 730810-1074418
【우편번호】 138-908
【주소】 서울특별시 송파구 잠실1동 주공아파트 34동 208호
【국적】 KR
【발명자】
【성명의 국문표기】 김범수
【성명의 영문표기】 KIM,Bum Soo
【주민등록번호】 690121-1019710

【우편번호】 431-080
【주소】 경기도 안양시 동안구 호계동 1055-1 무궁화아파트 703동 203호
【국적】 KR
【발명자】
【성명의 국문표기】 정태선
【성명의 영문표기】 CHUNG,Tae Sun
【주민등록번호】 710304-1030617
【우편번호】 156-034
【주소】 서울특별시 동작구 상도4동 279-400
【국적】 KR
【발명자】
【성명의 국문표기】 인지현
【성명의 영문표기】 IN,Ji Hyun
【주민등록번호】 771124-2063411
【우편번호】 120-090
【주소】 서울특별시 서대문구 홍제동 무악청구아파트 110동 802호
【국적】 KR
【우선권주장】
【출원국명】 KR
【출원종류】 특허
【출원번호】 10-2003-0021502
【출원일자】 2003.04.04
【증명서류】 미첨부
【심사청구】 청구
【취지】 특허법 제42조의 규정에 의한 출원, 특허법 제60조의 규정에 의한 출원심사 를 청구합니다. 대리인
김동진 (인)
【수수료】

【기본출원료】	20	면	29,000 원
【가산출원료】	33	면	33,000 원
【우선권주장료】	1	건	26,000 원
【심사청구료】	19	항	717,000 원
【합계】	805,000 원		

【첨부서류】 1. 요약서·명세서(도면)_1통 2.우선권증명서류 및 동 번역문_1통

【요약서】

【요약】

플래시 메모리의 사용 중에 발생하는 오류블록을 관리하는 방법 및 장치를 제공한다.

본 발명의 플래시 메모리의 오류블록 관리방법은 (a) 플래시 메모리에 복수의 사용블록을 갖는 사용영역과 복수의 예비블록을 갖는 예비영역을 할당하고, 상기 사용영역 또는 상기 예비영역에서 발생하는 오류블록과 이를 대체하여 사용하는 예비블록과의 매핑정보를 저장하는 블록맵페이지를 복수개 포함하는 블록맵페이지그룹을 상기 예비영역에 생성하는 단계와, (b) 상기 블록맵페이지그룹 중에 발생된 오류블록과 이를 대체하는 예비블록과의 매핑정보를 중 가장 최근의 매핑정보를 담고 있는 블록맵페이지("블록맵정보"라 함)를 주메모리에 상주시키는 단계, 및 (c) 플래시 연산 중에 오류블록이 발생할 경우에 상기 블록맵정보를 통해 미사용 예비블록을 찾고, 찾아진 미사용 예비블록에 발생된 상기 발생된 오류블록에 기록된 정보를 복원하여 기록하고, 상기 발생된 오류블록 및 상기 찾아진 예비블록과의 매핑정보를 바탕으로 상기 블록맵정보를 갱신하고 상기 블록맵페이지그룹 중의 어느 한 블록맵페이지에 상기 갱신된 블록맵정보를 기록하는 단계를 포함한다. 한편, 본 발명은 상기 (a) 단계에서 상기 사용영역의 사용블록들 중 일부와 상기 예비영역의 예비블록들 중 일부를 보호영역으로 지정하여 검증되지 않은 소프트웨어의 접근을 차단한다.

또한 본 발명의 플래시 메모리의 오류블록을 관리장치는 복수의 사용블록을 갖는 사용영역과 사용 중 발생하는 오류블록을 대체하기 위한 복수의 예비블록을 갖는 예비영역을 갖으며, 사용 중 발생되는 오류블록과 이를 대체하는 예비블록과의 매핑정보를 저장하는 블록맵페이지를 복수 개 갖고 있는 블록맵페이지그룹을 상기 예비영역 안에 갖고 있는 플래시 메모리와, 상기 블록맵페이지그룹 안의 가장 최신의 블록맵페이지 정보를 포함하며 오류블록이 발생될 때 이를

처리하기 위한 프로세스를 정의하는 플래시 장치관리자를 포함하는 주메모리, 및 상기 플래시 메모리와 상기 주메모리에 전기적으로 연결되어 있고 상기 주메모리의 플래시 장치관리자의 코드를 읽어 상기 플래시 메모리에 플래시 연산 및 오류블록 관리를 하는 중앙처리장치를 포함한다.

본 발명에 따라 최소의 플래시 연산으로 오류블록을 관리할 수 있다. 또한 중요한 데이터를 보호영역으로 설정하여 검증되지 않은 어플리케이션에 의한 악의적인 데이터 훼손이나 오동작에 의한 데이터 훼손을 방지할 수 있다.

【대표도】

도 12

【색인어】

플래시 메모리, Flash Memory, Bad Block, 오류블록, 매핑

【명세서】

【발명의 명칭】

플래시 메모리의 오류블록 관리방법 및 장치{METHOD AND APPARATUS FOR MANAGING BAD BLOCK IN FLASH MEMORY}

【도면의 간단한 설명】

도 1은 종전 발명의 플래시 메모리 시스템에서 오류블록 관리의 기본 단위인 청크(Chunk)의 구조를 보여주는 블록도이다.

도 2는 도 1에 도시된 청크맵을 보다 상세한 구조를 보여주는 블록도이다.

도 3은 본 발명의 시스템 구성요소를 간략히 보여주는 블록도이다.

도 4는 도 3의 시스템의 동작을 설명하기 위한 블록도이다.

도 5는 k+1개의 플래시 메모리 칩으로 구현한 플래시 메모리의 블록도이다.

도 6은 t개의 사용영역의 파티션과 한 개의 예비영역 파티션으로 나누어진 플래시 메모리의 어느 한 칩의 구조를 보여주는 블록도이다.

도 7은 비보호영역과 보호영역을 모두 포함하는 플래시 메모리의 어느 한 칩의 구조를 보여주는 블록도이다.

도 8은 모든 영역이 비보호영역으로 이루어진 플래시 메모리의 어느 한 칩의 구조를 보여주는 블록도이다.

도 9은 y개의 블록으로 나누어진 한 파티션의 구조를 보여주는 블록도이다.

도 10은 비보호영역과 보호영역을 모두 포함하는 예비영역의 구조를 보여주는 블록도이다.

도 11은 전 영역이 비보호영역으로 이루어진 예비영역의 구조를 보여주는 블록도이다.

도 12는 전 영역이 보호영역으로 이루어진 예비영역의 구조를 보여주는 블록도이다.

도 13은 블록맵페이지그룹의 구조를 상세히 보여주는 블록도이다.

도 14는 플래시 메모리에 대한 플래시 장치관리자의 동작 프로세스를 보여주는 흐름도이다.

도 15는 예비영역포맷 프로세스를 보다 상세히 보여주는 흐름도이다.

도 16은 예비영역포맷 프로세스 중 어느 한 칩의 예비영역포맷 프로세스를 보다 상세히 보여주는 흐름도이다.

도 17은 블록맵정보의 램상주 프로세스를 보다 상세히 보여주는 흐름도이다.

도 18은 어느 한 칩에 대한 블록맵정보의 램상주 프로세스를 보다 상세히 보여주는 흐름도이다.

도 19는 에러정정코드로 칩t의 읽기오류블록을 복구화하는 과정을 상세히 보여주는 흐름도이다.

도 20는 오류블록처리 프로세스를 보다 상세히 보여주는 흐름도이다.

도 21은 읽기오류블록처리 프로세스를 보다 상세히 보여주는 흐름도이다.

도 22은 쓰기/지우기오류블록처리 프로세스를 보다 상세히 보여주는 흐름도이다.

【발명의 상세한 설명】

【발명의 목적】

【발명이 속하는 기술분야 및 그 분야의 종래기술】

<23> 본 발명은 플래시 메모리 사용 중 발생되는 오류블록을 처리하는 방법 및 장치에 관한 것으로 보다 상세하게는 플래시 연산을 최소화하는 플래시 메모리 오류블록 관리 방법에 관한 것이다. 또한 본 발명은 검증되지 않은 플래시 어플리케이션에 의한 악의적인 플래시 연산으로부터 플

래시 메모리에 저장되어 있는 중요한 정보를 보호하기 위하여 플래시 메모리의 일정 영역을 보호영역으로 설정하고 관리하는 방법 및 장치에 관한 것이다.

<24> 플래시 메모리는 RAM이나 ROM 또는 하드디스크와 같이 데이터 저장수단이지만 플래시 메모리는 복수의 블록을 갖고있으며 블록단위로 지우기 연산을 한다는 점에서 특징을 갖는다.

<25> 플래시 메모리의 각 블록은 지우기 연산 횟수의 제한이 있다. 그래서 플래시 메모리가 공장에서 출하된 이후에 지우기 연산의 횟수가 제한을 넘게 되면, 해당 블록은 오류블록이 될 수 있으며, 또한 물리적 특성이 좋지 않는 블록인 경우, 지우기 연산 횟수가 제한 횟수에 접근하게 되면 해당 블록은 오류블록(Bad Block)이 될 수 있다. 그리고, NAND 플래시 메모리 같은 플래시 메모리는 공장에서 출하될 때부터 몇몇 블록은 오류블록일 수 있으며, 이때의 오류블록을 찾는 방법은 제조사에서 제공한다. 또한, 플래시 메모리에 읽기 연산을 할 때 이전에 플래시 메모리에 쓰였던 데이터와 다른 데이터가 읽힐 수도 있으며, 해당 블록 또한 오류블록이다. 이러한 오류블록들은 플래시 메모리를 사용 중에 모든 블록에서 발생할 수 있으며, 또는 사용하기 전부터 오류블록일 수 있다. 오류블록의 존재는 플래시 메모리 사용을 어렵게 만드는 요소이다.

<26> 이러한 오류블록을 관리하기 위해서는 오류블록을 관리하기 위한 정보가 필요하며 이런 정보를 RAM에 또는 EEPROM과 같은 별도의 메모리에 저장하여 관리할 수 있다. 그러나 RAM에 관리 정보를 저장했을 경우에는 전원이 꺼지면 RAM에 저장된 정보는 모두 잃어버리게 되는 문제점이 있고, EEPROM과 같은 별도의 메모리에 관리 정보를 저장하는 경우는 시스템 구축을 위한 추가 비용이 필요하고 시스템을 복잡하게 만들 수 있다.

<27> 이러한 문제점을 해결하기 위하여 미국특허 6,260,151 B1에 개시된 종전 발명은 플래시 메모리 시스템에서 오류블록(Bad Block)을 처리하는 방법을 제시하였는데 플래시 메모리의 일정한 영역을 오류블록 관리를 위한 부분으로 할당하여 오류블록(Bad Block)을 처리한다.

<28> 도1은 상기 발명의 오류블록 관리의 기본 단위인 청크(Chunk)의 구조를 보여주는 블록도이다. 종전 발명은 플래시 메모리 영역을 실제 데이터를 쓰고 지우는데 사용되는 데이터 영역과 오류블록이 발생된 경우에 이를 처리하기 위한 여분의 공간인 여분영역(Spare Area)으로 구분하고, 여분영역은 여러 개의 청크로 구분하였다. 도1은 청크의 구조를 보여주고 있다. 청크는 여분블록 0부터 여분블록 y까지의 y+1개의 여분블록으로 이루어져 있다. 청크 하나에는 메모리 번지수가 큰 영역에 오리지날청크맵(Original Chunk Map) 및 카피청크맵(Copy Chunk Map)이 각각 하나의 여분블록에 저장된다. 그리고 휘발성 메모리인 램에는 워킹청크맵(Working Chunk Map)이 존재한다.

<29> 도2는 청크맵의 보다 상세한 구조를 보여준다. 즉, 여분블록 0을 확대한 그림이다. 청크맵은 플래시 메모리에 존재하는 오리지날청크맵, 카피청크맵, 및 램에 존재하는 워킹청크맵 모두가 같은 데이터 구조를 갖는다. 우선 오류블록관리 헤더(BBM Header) 부분에는 오류블록을 관리하기 위한 메타 정보가 저장되는데 예를 들면, 총 블록의 수, 청크맵의 크기, 청크맵의 상태 등이 있다. 그리고 각각 유효필드(Valid Field)와 오류데이터블록번호(Bad Data Block Number)를 갖는 엔트리(BBM Entry)가 여분블록의 개수와 같은 y+1개가 있다. 엔트리는 오류블록과 이를 대체하는 여분블록의 매팽 정보가 저장된다. 예를 들어 7번 엔트리의 오류데이터블록번호가 10이라면 오류블록인 10번 블록이 청크 n의 7번 여분블록에 재매핑(Re-mapping)되어 있음을 나타낸다.

<30> 상기 종래기술의 동작은 다음과 같다. 먼저 오류블록 관리자를 초기화하는데, 오류블록 관리자의 초기화 과정은 오류블록 관리자포맷, 청크포맷, 청크마운트 과정으로 이루어지는데 오류블록 관리자의 초기화가 완료되는 시점은 청크가 마운트되는 시점이다. 청크를 마운트하는 과정은 플레시 메모리의 카피청크맵과 오리지날청크맵을 읽어서 램상에 워킹청크맵을 생성하는 과정이다. 이 때 플래시 메모리에 카피청크맵과 오리지날청크맵 두개를 유지하여 각종 연산 중에 오류가 발생하더라도 복구할 수 있는 방법을 제공한다. 오류블록 관리자 초기화 작업이 끝나면 준비 상태(Ready State)가 된다. 특정 블록에 대한 접근 방법은 각 청크에 대하여 요청된 블록이 엔트리에 존재하는지 검사한 후 만일 엔트리에 존재하면 재매핑한 결과를 보내준다. 새로운 오류블록이 발견되었을 때의 알고리즘을 요약하면 다음과 같다. 우선 사용할 수 있고 비사용 엔트리를 찾고 그 엔트리를 포함하는 청크맵을 읽어서 워킹청크맵에 쓴다. 다음으로 새로운 매핑정보를 워킹청크맵의 찾아진 비사용 엔트리에 쓴다. 그리고 나서 변경된 재매핑 정보를 플래시 메모리의 오리지날청크맵 및 카피청크맵에 기록한다.

<31> 상술한 종래 기술은 플래시 메모리상의 카피청크맵과 오리지날청크맵 및 램상의 워킹청크맵이 일관성을 유지하도록 여러 번의 플래시 메모리에 대한 쓰기와 지우기 연산을 수행한다. 예를 들어 새로운 실행시간 오류블록이 발생했을 경우 다음과 같은 연산을 필요로 한다.

<32> 1. 오리지날청크맵 블록 지우기 연산

<33> 2. 오리지날청크맵 블록에 오리지날청크맵 쓰기 연산

<34> 3. 오리지날청크맵의 상태를 무효(Invalid)로 쓰기 연산

<35> 4. 카피청크맵 블록 지우기 연산

<36> 5. 카피청크맵 블록에 카피청크맵 쓰기 연산

<37> 6. 오리지날청크맵의 상태를 유효(Valid)로 쓰기 연산

<38> 즉, 플래시 메모리에 두 번의 지우기 연산과 네 번의 쓰기 연산을 하게 된다. 또한 오류블록과 여분블록의 매핑 정보가 여러 개의 청크에 저장되기 때문에 오류블록을 처리하기 위해서 각 청크를 마운트하고 각 청크에 존재하는 청크맵간의 일관성을 유지하기 위하여 복잡한 알고리즘을 수행하여야 한다. 따라서 실행시간 오류블록이 발생하여도 플래시 연산의 횟수를 많이 필요로 하지 않는 발명이 필요하다.

<39> 최근에는 플래시 메모리를 단순히 데이터를 저장하기 위한 공간으로 사용하는 것을 넘어서 OS(Operating System) ROM 이미지나 부트로더(Boot Loader) 등 보호의 필요성을 갖는 데이터를 저장하는데 이용하는 경우가 생겼다. OS ROM 이미지나 부트로더처럼 시스템의 동작에 중요한 역할을 하는 데이터를 플래시 메모리에 저장하는 경우에 해당 영역을 플래시 메모리 관련 소프트웨어의 잘못된 동작이나 사용자의 실수에 의해 발생할 수 있는 지우기 또는 쓰기 연산으로부터 보호할 필요성이 있다. 즉, 플래시 메모리의 일정 영역을 읽기 전용 영역으로 설정하여 관리할 필요성이 있다. 한편 읽기 전용 영역이 설정된 플래시메모리의 경우에도 그 영역 안에서 발생되는 혹은 발생할 수 있는 오류블록을 대체하기 위한 방법이 필요하며 이 경우에는 오류블록을 대신하여 대체된 새로운 블록 또한 읽기 전용 블록으로 설정하여 관리할 필요성이 있다.

【발명이 이루고자 하는 기술적 과제】

<40> 본 발명의 목적은 오류영역이 발생했을 때 적은 플래시 연산의 횟수로 오류영역을 관리할 수 있는 방법 및 장치를 제공하는 것이다.

<41> 본 발명의 다른 목적은 플래시 관련 소프트웨어의 잘못된 동작이 플래시 메모리의 일정 영역의 데이터를 변경하는 것을 방지하기 위해서, 플래시 메모리의 일정 영역을 보호영역으로 관리하는 방법 및 장치를 제공하는 것이다.

<42> 본 발명의 또 다른 목적은 플래시 메모리의 일정 영역이 읽기 전용으로 보호되고 있는 경우에 이 영역에서 발생하는 오류블록에 대한 관리 방법 및 장치를 제공하는 것이다.

【발명의 구성 및 작용】

<43> 상기 목적을 달성하기 위하여 본 발명의 플래시 메모리의 오류블록 관리방법은 (a) 플래시 메모리에 복수의 사용블록을 갖는 사용영역과 복수의 예비블록을 갖는 예비영역을 할당하고, 상기 사용영역 또는 상기 예비영역에서 발생하는 오류블록과 이를 대체하여 사용하는 예비블록과의 매핑정보를 저장하는 블록맵페이지를 복수개 포함하는 블록맵페이지그룹을 제공하는 단계와, (b) 상기 블록맵페이지그룹 중에서 소정의 규칙에 의해 선택된 블록맵페이지의 매핑정보를 메모리에 상주시키는 단계, 및 (c) 플래시 연산 중에 발생된 오류블록을 상기 상주된 매핑정보를 통해 찾은 미사용 예비블록과 매핑하고, 상기 상주된 매핑정보를 갱신하고 상기 갱신된 매핑정보를 상기 블록맵페이지그룹에 속하는 블록맵페이지에 기록하는 단계를 포함한다.

상기 (a)단계는 플래시 메모리에 복수의 사용블록을 갖는 사용영역과 복수의 예비블록을 갖는 예비영역을 할당하는 (a1) 단계와, 상기 생성된 예비블록과 발생될 오류블록의 매핑정보를 기록할 블록맵필드를 예비블록마다 생성하고, 상기 생성된 블록맵필드들, 매핑정보의 시점을 판단하기 위한 카운트 필드, 및 유효성을 판단을 위한 트랜지션 필드를 포함하는 블록맵정보를 초기화시키는 (a2) 단계와, 플래시 메모리에 존재하는 오류블록을 검사하여 발견된 오류블록들 및 상기 발견된 오류블록들을 대체하는 예비블록들과의 매핑정보를 상기 블록맵필드들에 각각 기록하고 상기 카운트 필드에 초기 카운트를 하여 블록맵정보를 생성하는 (a3) 단계, 및 상기

블록맵정보를 상기 블록맵페이지그룹의 어느 한 블록맵페이지에 기록하는 (a4) 단계를 포함한다. 상기 블록맵페이지그룹은 최소한 두 개의 예비블록으로 구성되는 것이 바람직하다. 상기 플래시 메모리가 복수의 칩으로 이루진 경우에 각 단계들은 각각의 칩마다 수행되는 것이 바람직하다. 또한, 상기 (a) 단계에서 상기 사용블록들 및 상기 예비블록들 중 적어도 일부를 보호영역으로 지정하여 검증되지 않은 소프트웨어의 접근을 차단하는 것이 바람직하다.

<44> 상기 (b) 단계는 블록맵페이지그룹 안의 블록맵페이지들 중에서 유효하면서 가장 최근에 기록된 블록맵페이지를 검색하여 검색된 블록맵페이지의 매핑정보를 메모리에 상주 시킨다. 상기 유효하면서 가장 최근에 기록된 블록맵페이지를 검색할 때 각각의 블록맵페이지들의 트랜지션 필드에 기록된 값 및 카운트 필드에 기록된 숫자의 크기를 기준으로 검색하는 것이 바람직하다. 상기 블록맵페이지그룹 안의 블록맵페이지들 중에서 유효하면서 가장 최근의 블록맵페이지를 검색하는 과정에서 유효하지 않으면서 읽기오류가 표시된 블록맵페이지를 발견할 경우, 상기 발견된 읽기 오류블록을 소정의 방법으로 복구하는 단계를 더 포함하는 것이 바람직하다. 상기 단계들은 플래시 메모리가 복수의 칩으로 이루어진 경우에 각 단계들은 칩마다 수행되는 것이 바람직하다.

<45> 상기 (c) 단계는 상기 상주된 매핑정보를 이용하여 미사용 예비블록을 찾는 (c1) 단계와, 상기 (c1) 단계를 통해 찾아진 미사용 예비블록과 오류블록의 번호를 매핑하여 상기 상주된 매핑정보를 갱신하는 (c2) 단계와, 상기 블록맵페이지 그룹의 미사용 블록맵페이지를 찾는 (c3) 단계와, 상기 발생된 오류블록에 기록된 정보를 상기 찾아진 미사용 예비블록에 복사하는 (c4) 단계와, 상기 갱신된 매핑정보를 포함한 정보를 상기 찾아진 미사용 블록맵페이지에 기록하는 (c5) 단계, 및 상기 미사용 블록맵페이지가 유효함을 표시하는 (c6) 단계를 포함한다. 상기 (c) 단계는 상

기 오류블록이 보호영역에서 발생된 경우에 상기 보호영역을 읽기 쓰기 가능 상태로 변경하고, 상기 (c6) 단계가 끝난 후에 상기 보호 영역을 다시 읽기 전용 상태로 변경하는 단계를 더 포함하는 것이 바람직하다. 상기 각 단계들은 플래시 메모리가 복수의 칩으로 이루어진 경우에 각 단계들은 칩마다 수행되는 것이 바람직하다.

<46> 한편, 상기 (c) 단계는 읽기 오류블록이 발생된 경우에 이를 복구하기 위하여 발생된 오류블록이 읽기 오류블록인지 여부를 판단하는 (c1) 단계와, 읽기 오류블록을 복구하기 위해 임시로 사용하는 예비블록을 지우는 (c2) 단계와, 읽기 오류블록의 오류를 정정하여 상기 예비블록에 복사하는 (c3) 단계와, 상기 블록맵페이지그룹 중에 미사용 블록맵페이지를 찾는 (c4) 단계와, 상기 찾아진 블록맵페이지에 상기 읽기 오류블록과 상기 예비블록의 매핑정보 및 읽기오류 복구 상태임을 표시하는 (c5) 단계와, 상기 읽기 오류블록을 지우는 (c6) 단계와, 상기 예비블록을 상기 지워진 읽기 오류블록에 복사하는 (c7) 단계, 및 상기 찾아진 블록맵페이지에 무효임을 표시하는 (c8) 단계를 포함한다. 상기 읽기 오류블록이 보호영역에서 발생된 경우에 보호영역을 읽기 쓰기 가능 상태로 변경하고, 상기 (c8) 단계가 끝난 후에 상기 보호 영역을 다시 읽기 전용 상태로 변경하는 것이 바람직하다.

<47> 상기 목적달성을 위하여 본 발명에 따른 최소한 하나 이상의 플래시 메모리 칩으로 이루어진 플래시 메모리의 오류블록을 관리장치는 복수의 사용블록을 갖는 사용영역과 사용 중 발생하는 오류블록을 대체하기 위한 복수의 예비블록을 갖는 예비영역을 가지며, 사용 중 발생되는 오류블록과 이를 대체하는 예비블록과의 매핑정보

를 저장하는 블록맵페이지를 복수 개 갖고 있는 블록맵페이지그룹을 상기 예비영역 안에 갖고 있는 플래시 메모리와, 상기 블록맵페이지그룹 안에 존재하는 최근의 유효한 블록맵페이지의 매핑정보와, 오류블록이 발생될 때 이를 처리하기 위한 프로세스를 정의하는 플래시 장치관리자를 로딩하기 위한 메모리, 및 상기 플래시 메모리와 상기 메모리에 전기적으로 연결되어 있고 상기 플래시 장치관리자의 코드를 읽어 상기 플래시 메모리에 대한 플래시 연산 및 오류블록 관리를 하는 중앙처리장치를 포함한다.

<48> 상기 블록맵페이지그룹은 최소한 2개 이상의 예비블록으로 구성된 것이 바람직하다. 한편, 상기 중앙처리장치와 상기 플래시 메모리 사이에 전기적으로 연결되어 있으며 상기 중앙처리장치가 플래시 연산을 하는 동안 플래시 메모리에 다음 연산을 위한 데이터를 저장하고 있는 버퍼를 두 개 이상 가지고 있는 플래시 메모리 콘트롤러를 더 포함하는 것이 바람직하다. 상기 플래시 메모리 콘트롤러는 상기 플래시 메모리의 일정영역을 보호영역으로 설정하여 상기 플래시 장치관리자에 의해 검증된 플래시 어플리케이션에 대해서만 상기 보호영역에 대한 플래시 연산을 허용하고 검증되지 않은 플래시 어플리케이션에 대해서는 상기 보호영역에 대한 플래시 연산을 허용하지 않는 것이 바람직하다.

<49> 상기 플래시 메모리가 복수의 플래시 메모리 칩으로 이루어진 경우에 상기 플래시 메모리의 사용영역, 예비영역과 상기 예비영역 안의 블록맵페이지그룹은 각 플래시 메모리 칩마다 존재하는 것이 바람직하다.

<50> 이하, 첨부도면을 참조하여 본 발명에 따른 바람직한 실시예를 상세히 설명한다.

<51> 도 3은 본 발명을 구현하는 시스템의 일 실시예를 보여주고, 도 4는 시스템의 동작을 보여준다.

<52> 시스템은 플래시 메모리(100), 플래시 메모리 콘트롤러(200), 중앙처리장치(300), 및 XIP(Execute-In-Place)가능한 메모리(400)를 포함한다. XIP(Execute-In-Place)가능한 메모리(400)란 램처럼 그 안에서 소프트웨어를 실행시킬 수 있는 메모리(이하, "주메모리"라고 함)를 말한다.

<53> 중앙처리장치(300)는 주메모리(400)에 상주한 플래시 어플리케이션(410) 및 플래시 장치관리자(420)를 이용하여 플래시 메모리(100)에 필요한 정보를 쓰고 읽거나 지울 수 있으며(이하, "플래시 연산"이라고 함), 지우기 연산은 블록단위로 수행해야 한다.

<54> 도 3과 같이 플래시 메모리 콘트롤러(200)가 존재하는 시스템에서는, 플래시 어플리케이션(410)은 항상 플래시 메모리 콘트롤러(200)을 통해서 플래시 메모리(100)에 접근한다. 현재 플래시 메모리(100)는 읽기 연산의 속도에 비해 쓰기 연산의 속도가 약 20배에서 1000배 정도 느리며, 플래시 메모리 콘트롤러(200)는 쓰기 또는 읽기 연산의 속도를 향상 시킬 수 있는데, 이를 위해서 플래시 메모리 콘트롤러(200)의 내부에 쓰기 또는 읽기 연산을 위한 버퍼를 두 개 이상을 둔다. 그리고, 중앙처리장치(300)가 플래시 메모리 콘트롤러(200)의 버퍼에 데이터를 입력 또는 출력하는 동안 플래시 메모리 콘트롤러(200)는 다른 버퍼의 데이터를 플래시 메모리(100)로 출력 또는 입력 함으로써, 플래시 메모리 콘트롤러(200)는 중앙처리장치(300)에서 플래시 메모리(100)로의 데이터 입출력 속도를 향상시킬 수 있다.

<55> 플래시 어플리케이션(410)이 플래시 메모리(100)을 이용하기 위해, 플래시 메모리 콘트롤러(200)를 접근할 때, 플래시 메모리 콘트롤러(200)는 플래시 메모리를 일부 영역을 읽기전용영역으로 설정하여, 플래시 어플리케이션(410)의 잘못된 플래시 연산으로부터 데이터를 보호할 수 있다. 읽기전용영역에 대한 사용을 위해 플래시 메모리 콘트롤러(200)는 검증된 플래시 어플리케이션(412)임을 인증하고 인증을 통과한 검증된 플래시 어플리케이션(412)만이 플래시 메

모리(100)의 읽기전용영역에 접근하여 쓰기 및 지우기 연산을 할 수 있으며, 인증을 받지 못한 비검증된 플래시 어플리케션으로(414)은 플래시 메모리(100)의 읽기전용영역에 접근하여 쓰기 및 지우기 연산을 할 수 없도록 하여 잘못된 플래시 연산으로부터 데이터를 보호할 수 있다.

<56> NAND 플래시 메모리와 같은 I/O 버스 인터페이스를 통해 제어할 수 있는 플래시 메모리는 일반적으로 중앙처리장치(300)의 메모리 버스 인터페이스에 연결 또는 제어하기가 쉽지 않다. 그래서, 플래시 메모리 콘트롤러(200)는 I/O 버스 인터페이스를 갖는 플래시 메모리(100)와 중앙처리장치(300)의 사이에 위치하여 서로 상이한 버스 인터페이스를 연결을 도와줌으로써, 쉽게 플래시 메모리(100)를 사용할 수 있게 한다. 플래시 메모리 콘트롤러(200)는 플래시 메모리(100)의 일정 영역을 읽기 전용 영역으로 만들거나 플래시 연산성능을 향상시키는 등의 여러 가지 기능을 제공하나 플래시 메모리 콘트롤러(200)를 제외한 중앙처리장치(300)와 주메모리(400) 및 플래시 메모리(100)만으로도 시스템을 구현할 수 있다.

<57> 주메모리(400)에는 플래시 장치관리자(420)과 플래시 어플리케이션(410)이 상주할 수 있다. 플래시 장치관리자(420)는 플래시 어플리케이션(410)이 플래시 메모리(100)를 사용하다가 발생된 오류블록을 관리하는 모든 프로세스를 정의하는 소프트웨어로 구현한 모듈로서, 주메모리(400)에 상주하며 중앙처리장치(300)가 플래시 장치관리자(420)의 코드를 실행시킴으로써 오류블록을 관리한다. 플래시 어플리케이션(410)은 검증된 플래시 어플리케이션(412)과 비검증된 플래시 어플리케이션(414)으로 나눌 수 있다. 검증된 플래시 어플리케이션(412)은 플래시 장치관리자(420)를 사용하는 상위계층이며 플래시 메모리의 데이터를 고의적으로 파괴하는 잘못된 동작을 하지 않는 플래시 어플리케이션(410)이며 부트로더(Boot-loader), 플래시 트랜슬레이션 레이어(Flash Translation Layer), 파일 시스템 등이 될 수 있다. 비검증된 플래시 어플리케이션(414)은 시스템의 비정상적으로 동작시키기 위해 크래커 등에 의해서 만들어진 바이러

스 프로그램이나 해킹프로그램 등이 될 수 있으며, 플래시 메모리의 데이터를 고의적으로 파괴하는 잘못된 동작을 하는 플래시 어플리케이션(410)이다. 플래시 장치관리자(420)는 검증된 플래시 어플리케이션(412)이 플래시 메모리(100)를 사용할 수 있게 해준다.

<58> 도 4는 플래시 어플리케이션(412), 플래시 장치 관리자(420)와 플래시 메모리 콘트롤(200)사이에 반물리주소(semi-physical address)와 물리주소(physical address)에 관한 연관관계를 나타낸다. 검증된 플래시 어플리케이션(412)은 플래시 장치 관리자(420)에서 반물리주소에 플래시 연산을 요청하며, 플래시 장치 관리자(420)는 반물리주소를 물리주소로 변환하여 플래시 메모리 콘트롤러(100)에 플래시 연산을 한다. 반물리주소와 물리주소는 블록번호 * 블록의 크기 + 블록의 오프셋로 표현될 수 있다. 블록의 오프셋은 0보다 크거나 같으며 블록의 크기보다 작은 값이다.

<59> 반물리주소에서 블록번호는 항상 사용영역(110)의 예비블록번호 중 하나이다. 하지만 물리주소의 경우, 블록번호는 사용영역(120)의 사용블록이 오류블록인 경우, 예비영역의 예비블록번호 중 하나가 된다. 대부분의 경우에 있어서 반물리주소와 물리주소는 동일하며, 예비블록이 불량블록일 경우에만 다르게 된다.

<60> 한편 비검증된 플래시 어플리케이션(414)은 플래시 장치 관리자(420)를 통하지 않고 플래시 메모리 콘트롤러(200)에 접근하는 경우가 발생할 수도 있는데, 플래시 메모리 콘트롤러(200)에 검증된 플래시 어플리케이션(412)임을 인증하여 비검증된 플래시 어플리케이션(414)이 보호영역(130)에 플래시 연산을 할 경우 인증이 되지 않았기 때문에 플래시 연산은 허용되지 않는다.

<61> 도 5는 k+1개의 플래시 메모리 칩으로 구현한 플래시 메모리의 블록도이다. 플래시 메모리(100)는 k+1개의 플래시 메모리 칩으로 구현할 수 있는데 각 메모리 칩은 사용영역(110)과 오류블록을 관리하기 위한 예비영역(120)을 가질 수 있다. 물론 본 발명에서는 오류블록을 관리

하기 위한 예비영역을 특정 칩, 예를 들어 칩0, 칩1, 칩2에 몰아 넣을 수도 있다. 바람직한 실시예에 있어서 각 칩이 예비영역(120)을 갖도록 함으로써 플래시 연산을 위해 항상 오류블록 유무를 검사해야 하는 범위를 하나의 칩에 한정함으로써 플래시 연산의 성능을 높일 수 있다. 예비영역(120)의 크기는 각 칩에서 발생할 수 있는 최대 오류블록수와 오류블록을 관리하기 위해 필요한 블록수의 합으로 결정된다. 예비영역(120)에서 필요한 블록의 수를 결정하는 방법은 후술한다.

<62> 도 6은 t개의 사용영역의 파티션과 한 개의 예비영역의 파티션으로 나누어진 플래시 메모리 칩을 보여주는 블록도이다. 플래시 메모리에 부트로더롬이미지(Boot-Loader ROM Image), OS롬이미지(Operating System ROM Image), 또는 파일시스템 데이터 등을 저장하는데 있어 각각의 용도를 위해서 플래시 메모리에 영역을 구분하여 할당할 수 있고, 이렇게 구분된 영역을 파티션이라고 한다. 각 파티션은 파티션관리소프트웨어가 있으며 각 파티션관리소프트웨어는 자기의 파티션에서 발생할 수 있는 오류블록을 플래시 장치 관리자(420)를 통해 포괄적으로 관리한다. 파티션관리소프트웨어는 검증된 플래시 어플리케이션(412)이며, 부트로더, OS, 파일시스템 등이 될 수 있다. 사용영역(110)은 해당 칩의 최하위주소부터 할당되고, 예비영역(120)은 사용영역(110)이 사용하지 않는 곳에 할당되나, 이 순서는 바꿀 수도 있다. 이 때 사용영역(110)은 t개의 파티션으로 나눌 수 있는데, 파티션의 개수 t 및 크기는 플래시 메모리에 저장되는 정보의 종류와 량에 따라 결정된다.

<63> 도 7은 비보호영역과 보호영역을 모두 포함하는 플래시 메모리의 어느 한 칩의 구조를 보여주는 블록도이다. 플래시 메모리(100) 자체에서 또는 플래시 메모리 콘트롤러(200)가 플래시 메모리의 일정 영역을 읽기전용 영역(보호영역)으로 설정할 수 있다면 도7과 같은 구조로 해당 칩의 사용영역(110)과 예비영역(120)은 보호영역(130)과 비보호영역(140)으로 나눌 수 있다.

보호영역(130)은 검증된 플래시 어플리케이션(412)에게만 플래시 연산이 허용되며, 비검증 플래시 어플리케이션(414)은 플래시 연산은 허용되지 않거나 읽기 연산만 가능하다. 즉, OS롬이미지나 부트로더처럼 특별히 보호되어야할 데이터가 기록되는 영역이라고 할 수 있다. 플래시 어플리케이션이 보호영역(130)을 접근할 때는 항상 플래시 어플리케이션의 인증 단계를 거쳐야 한다. 플래시 어플리케이션이 검증된 플래시 어플리케이션인지 여부를 판단할 수 있는 인증 방법은 인증을 위한 특정의 코드 유무로 판단할 수 있다. 즉 특정의 코드가 있는 플래시 어플리케이션에 대해서 플래시장치관리자(420)는 플래시 메모리 콘트롤러(200)에 플래시 어플리케이션의 접근 허용을 요청할 수 있다. 그러나 이러한 방법이외에도 인증방법은 다른 여러 가지 방법으로 구현할 수 있다. 비보호영역(140)은 플래시 어플리케이션이(410) 읽기, 쓰기, 및 지우기가 모두 가능한 영역을 말한다. 도 7에서는 비보호영역(140)는 사용영역(110)과 예비영역(120)에 걸쳐 있는 형태로 구현하였다. 보호사용영역은 최하위주소부터 할당되고 그 위에 비보호사용영역, 비보호예비영역, 및 보호예비영역이 차례로 할당된다. 그러나 이 순서는 달리할 수도 있다.

<64> 도 8은 모든 영역이 비보호영역으로 이루어진 플래시 메모리의 어느 한 칩의 구조를 보여주는 블록도이다. 만일 OS롬이미지나 부트로더처럼 특별히 보호되어야할 데이터가 없는 경우라면 모든 영역을 비보호영역(140)으로 설정하여 사용할 수도 있다. 이런 경우에 모든 파티션에 대하여 플래시 연산이 가능하다.

<65> 도 9는 y개의 블록으로 나누어진 한 파티션의 구조를 보여주는 블록도이다. 블록은 플래시 메모리에서 지우기 연산의 단위로서 어떤 블록에 하나의 비트라도 오류가 생기면 해당 블록은 오류블록이 된다. 이러한 오류블록은 사용영역(110)에서 발생하기도 하지만 예비영역(120)에서도 발생할 수 있다.

<66> 도 10은 비보호영역과 보호영역을 모두 포함하는 예비영역의 구조를 보여주는 블록도이다. 도 10의 예비영역(120)은 비보호영역(140)의 오류블록을 대체하기 위한 m개의 예비블록(154)와 보호영역(130)의 오류블록을 대체하기 위한 n개의 예비블록(155)을 포함하여 총 m+n개의 예비블록과, 사용영역(110) 또는 예비영역(120)의 오류블록과 이를 대체하기 위한 비보호 또는 보호예비블록(154 또는 155)을 매핑하는 정보를 담고 있는 비보호 또는 보호블록맵페이지그룹(150 또는 158)과, 읽기영역의 오류를 처리하기 위하여 지정한 하나의 예비블록인 트랜스퍼블록(152), 및 파티션정보를 담고 있는 파티션정보블록(156)을 포함하고 있다. 상기 n과 m의 크기는 고정할 수도 있으나 변경가능하도록 구현할 수 있다. 예를 들면, 총 m+n=100 일 때, 보호블록에서 오류가 발생하지 않는 동안에 100번 연속해서 비보호 블록에서 오류가 발생한 경우에 m=100, n=0이 될 수 있다.

<67> 블록맵페이지그룹은 사용영역(110) 및 예비영역(120) 안의 비보호영역(140)의 오류블록과 이를 대체하는 예비블록(154)과의 매핑정보를 담고 있는 비보호블록맵페이지그룹(150)과, 사용영역(110) 및 예비영역(120) 안의 보호영역(130)의 오류블록과 이를 대체하는 예비블록(155)과의 매핑정보를 담고 있는 보호블록맵페이지그룹(158)을 포함하는 개념으로 사용한다. 비보호블록맵페이지그룹(150)과 보호블록맵페이지그룹(158)은 각각 2개의 블록으로 구성되는 것이 바람직하며 이에 대해서는 후술한다. 한편, 비보호블록맵페이지그룹(150)과 보호블록맵페이지그룹(158)을 구별하지 않고 하나의 블록맵페이지그룹으로 사용할 수도 있는데, 이 경우에도 블록맵페이지그룹은 두 개의 블록으로 구성되는 것이 바람직하다.

<68> 파티션정보블록(156)은 칩의 파티션에 관한 정보를 담고 있는 블록으로서 파티션에 대한 정보를 플래시 메모리에 저장하지 않는다면 파티션정보블록(156)은 예비영역(120)에 할당되지 않을 것이다.

<69> 트랜스퍼블록(152)는 읽기 오류가 발생한 블록을 회복하기 위해서 사용하는 블록이다. 플래시 메모리에서 읽기 오류가 발생한 경우에 지우기 연산을 하고 나면 다시 사용할 수 있는 경우가 있으므로 읽기 오류가 발생한 블록을 재사용하기 위해서 트랜스퍼블록을 사용할 수 있다. 만일 읽기오류를 쓰기 오류 및 지우기 오류와 동일하게 취급하여 오류블록을 예비블록으로 대체할 경우 트랜스퍼블록(152)은 불필요하다.

<70> 블록맵페이지그룹(150 또는 158), 트랜스퍼블록(152)과 파티션정보블록(156)은 오류블록이 아닌 정상블록에 할당되어야 한다. 현재 상용화된 플래시 메모리의 지우기 연산 가능횟수는 대략 1만에서 10만 정도이고 플래시 칩의 총 블록 개수는 대략 1만개 미만이다. 그리고, 플래시 메모리를 사용 중에 발생할 수 있는 오류블록의 개수는 플래시 칩의 총 블록 개수의 수 퍼센트이다. 오류블록이 발생 할 때, 블록맵페이지그룹(150 또는 158) 또는 트랜스퍼블록(152)에 지우기 연산을 할 수 있는데, 이때 지우기 연산의 총 횟수는 플래시 메모리에서 발생할 수 있는 오류블록의 개수와 같거나 작다고 할 수 있다. 이 때문에 블록맵페이지그룹(150 또는 158) 또는 트랜스퍼블록(152)를 위해 할당된 블록들이 오류블록이 될 확률은 극히 작으며, 따라서 블록맵페이지그룹(150 또는 158)이나 트랜스퍼블록(152) 혹은 파티션정보블록(156)의 오류에 대한 대비로는 최초에 오류블록이 아닌 정상블록으로 할당하는 것만으로 충분하다.

<71> 도 11은 전 영역이 비보호영역으로 이루어진 예비영역의 구조를 보여주는 블록도이다. 이 때, n은 0이 되고 m은 플래시 메모리를 사용하는 동안에 발생해도 플래시 메모리의 성능이 보장되는 최대 오류블록의 수와 같다. 도 12는 전 영역이 보호영역(130)으로 이루어진 예비영역(120)의 구조를 보여주는 블록도이다. 도 12에서는 사용영역(110) 또한 보호영역(130)이다. 이 때, m은 0이 되고 n은 플래시 메모리를 사용하는 동안에 발생해도 플래시 메모리의 성능이 보장되는 최대 오류블록의 수와 같다.

<72> 도 13은 블록맵페이지그룹(150 또는 158)의 구조를 상세히 보여주는 블록도이다. 블록맵페이지그룹은 q개의 블록맵페이지(160)로 구성되어 있고, 각 블록맵페이지(160)는 오류블록번호(166)와 이를 대체하는 예비블록번호(167)로 구성되는 블록맵필드(161)를 예비블록(154 또는 155)의 개수와 같은 m+n개를 갖는다. 즉, 하나의 블록맵페이지(160)는 오류블록과 이를 대체하는 예비블록과의 매핑정보의 모두를 포함한다. 또한 블록맵페이지(160)는 가장 최근에 쓰여진 블록맵페이지를 찾는데 쓰이는 카운트필드(163)와, 자신이 갖고 있는 블록맵필드정보(161)가 유효한지, 무효인지, 아니면 사용되지 않고 있는지를 구별할 수 있도록 하는 트랜지션필드(165)도 포함한다. 가장 최근에 씌여진 블록맵페이지(160)는 트랜지션필드(165)에 유효표시를 갖는 블록맵페이지(160) 중에서 카운트필드(163)에 가장 큰 숫자가 씌여진 것으로서, 이것을 주메모리, 예를 들어 램에 올린 것을 블록맵정보(170)라고 한다. 블록맵정보(170)를 통한 블록맵페이지그룹의 갱신은 후술한다. 한편, 본 발명의 바람직한 실시예에 있어서 오류블록번호(166)과 예비블록번호(167)은 모두 2바이트를 갖는데, 이는 현재 상용화된 플래시 메모리의 총 블록 개수는 1만개 미만이다. 따라서 2바이트는 0에서 65535까지의 숫자를 가질 수 있으므로 그 정도면 충분한 양이기 때문이다. 또한 본 발명의 바람직한 실시예에 있어서 카운트필드(163)는 2바이트를 갖는데, 현재 상용화된 플래시 메모리의 총 블록 개수는 1만개 미만이며, 플래시 메모리를 사용 중에 발생할 수 있는 오류블록의 개수는 플래시 칩의 총 블록 개수의 수 퍼센트이다. 따라서 2바이트는 0에서 65535까지의 숫자를 가질 수 있으므로 그 정도면 충분한 양이기 때문이다.

<73> 도 3의 주메모리(400)에 있는 플래시 어플리케이션 중에 검증된 플래시 어플리케이션(412)은 사용영역(110)의 특정한 사용블록을 접근하기 위해 플래시_장치관리자(420)에게 반물리주소를 보내고 플래시 장치 관리자(420)는 반물리주소에 맞는 물리주소를 갖는 블록을 찾는다. 이 때

오류블록을 사용하지 않도록 하기 위해서 가장 최근의 오류블록정보를 담고 있는 블록페이지(160)의 블록맵정보(170)에 등록된 블록맵필드(161) 중 사용블록번호가 오류블록넘버(166)와 일치하면 이를 대체하는 예비블록(167)을 사용한다.

<74> 도 14는 플래시 메모리에 대한 플래시 장치관리자의 동작 프로세스를 보여주는 흐름도이다. 도 15, 16은 예비영역포맷 프로세스를 보다 상세히 보여주는 흐름도이고, 도 17, 18, 19는 블록맵정보의 램상주 프로세스를 보다 상세히 보여주는 흐름도이며, 도 20, 21, 22은 오류블록처리 프로세스를 보다 상세히 보여주는 흐름도이다.

<75> 맨 처음 플래시 메모리의 전 영역을 읽기 쓰기 가능으로 설정(S90)한 후, 각 칩을 사용영역과 예비영역을 나누고 예비영역에 블록맵페이지그룹을 만드는 프로세스, 즉 예비영역을 포맷하는데(S100), 상기 S90 및 S100은 처음 1회 때만 필요한 프로세스이다. 예비영역포맷(S100)의 동작은 도 15 및 도 16을 참조하여 설명한다.

<76> 예비영역의 포맷(S100)은 플래시 칩 번호 t를 0으로 초기화(S110)한다. 그리고, 칩t(t=0)의 예비영역 포맷(S130)을 실행한 후에, 마지막 칩인지를 검사(S170)한다. 마지막 칩이 아니라면, 플래시 칩 번호 t를 1증가(S120)하고 다음 칩에 대한 포맷을 실행한다. 그리고 모든 칩에 대한 포맷이 끝나면 예비영역포맷이 종료된다.

<77> 칩t의 예비영역을 포맷(S130)하기 위해서는 먼저 칩의 일정부분은 사용영역으로 할당하고 일정영역은 플래시 메모리 사용 중에 발생하는 오류블록에 대한 대비책으로 칩t의 예비영역으로 할당한다(S132). 예비영역을 위하여 할당해야 할 블록의 개수는 플래시 메모리 사용 중에 발생할 수 있는 최대 오류블록의 개수와 발생되는 오류블록을 관리하기 위한 블록의 개수의 합으로 구할 수 있다. 전자에 할당된 블록은 예비블록들의 모임이고, 후자에 할당된 블록은 블록맵페이지그룹

트랜스퍼블록과 파티션정보블록이 이에 해당한다. 오류블록을 관리하기 위한 블록의 숫자는 다음과 같이 할당할 수 있다. 먼저 칩을 보호영역과 비보호영역으로 나누어 사용할 경우에 보호블록맵페이지그룹과 비보호블록맵페이지그룹을 위하여 각각 2개의 블록을 할당하며, 칩을 비보호영역만으로 사용할 경우에는 비보호블록맵페이지그룹을 위한 2개의 블록만을 할당한다. 각 블록맵페이지그룹에 2개의 블록을 할당하는 이유는 후술한다.

<78> 파틴션 정보를 칩에 저장하고자 하는 경우에는 파티션정보블록을 한 블록 할당한다. 또한 읽기 오류가 발생한 블록을 예비블록으로 대체하지 않고 재사용할 수 있도록 하는데 필요한 트랜스퍼 블록도 한 블록을 할당한다. 이를 표로 정리하면 다음과 같다.

<79> 【표 1】

오류블록관리를 위하여 할당될 총 블록 수

비보호/보호영역 지원	비보호영역 지원	트랜스퍼 블록 지원	파티션 정보블록 지원	할당될 총블록수
0		0	0	6
0		0		5
0			0	5
0				4
	0	0	0	4
	0	0		3
	0		0	3
	0			2

<80> 칩t의 예비영역할당(S132)이 끝나면, 칩t의 블록맵정보를 초기화한다(S134). 각 칩의 블록맵정보는 평상시에는 도 1의 주메모리(400)에 상주하는 플래시 장치 관리자에 포함된 정보로서 원칙적으로 각 칩의 블록맵페이지그룹에 저장되어 있는 가장 최근의 블록맵페이지와 동일한 정보를 갖는다. 맨 처음 플래시 메모리를 사용할 때는 블록맵정보의 생성을 위하여 블록맵정보를 초기화해야 한다. 칩t의

_블록맵정보의 초기화(S134)는 블록맵정보의 크기를 결정하고 각 필드를 특정 값으로 초기화하는 것을 의미한다. 블록맵정보의 크기는 트랜지션필드(165)크기 + 카운트필드(163)크기 + 블록맵필드 한개의 크기*예비블록의 개수로 정해진다. 각 필드의 초기값을 살펴보면, 트랜지션필드(165)는 유효표시를 갖고, 카운트필드(163)은 최초로 기록될 블록맵정보라는 의미에서 1을 갖는다. 그리고, 블록맵필드(161)의 오류블록번호(166)과 예비블록번호(167)은 다음과 같이 초기화한다. 오류블록번호(166)와 예비블록번호(167)에서 표현할 수 있는 최대값을 각 블록맵필드(161)의 오류블록번호(166)과 예비블록번호(167)에 기록함으로써, 모든 블록맵필드(161)를 미사용 블록맵필드로 만든다. 본 발명에서 오류블록번호(166)과 예비블록번호(167)를 위한 기록하기 위한 공간은 바람직한 실시예는 2바이트인데, 이는 현재 상용화된 플래시 메모리의 총 블록 개수는 1만개 미만이기 때문이다. 즉, 각 블록번호(166 또는 167)가 가질 수 있는 최대 값은 65535이며, 65535은 미사용블록의 개수를 나타내기에 충분하다.

<81> 칩t의 블록맵정보가 초기화되면(S134) 이를 바탕으로 칩t의 블록맵정보를 생성한다(S136). 블록맵정보를 생성하려면 칩t에서 사용영역(110)의 최하위주소부터 최상위주소까지 오류블록이 존재하는 지를 검사한다. 오류블록이 발견되면 칩t의 블록맵필드에서 미사용 블록맵필드를 찾고, 상기 찾은 미사용 블록맵필드의 오류블록번호(166)과 예비블록번호(167)에 발견된 오류블록과 이를 대체하는 예비블록의 번호를 적는다. 미사용 블록맵필드를 찾을 때는 블록맵필드1(161)부터 블록맵필드m+n(161)까지 순차적으로 검사하여 최초로 발견된 미사용 블록맵필드를 취하는 것이 바람직하다.

<82> 한편, 플래시 메모리를 보호영역과 비보호영역으로 나누어 사용하는 경우에 각각의 블록맵정보(보호블록맵정보와 비보호블록맵정보)를 생성할 수도 있지만 하나의 블

록맵정보로 생성할 수도 있다. 물론 보호영역과 비보호영역으로 나누더라도 블록맵페이지그룹을 하나로 통합해서 사용하는 경우에는 블록맵정보를 하나로 생성할 수 있다.

<83> 마지막으로 생성된 블록맵정보를 칩t의_예비영역의 블록맵페이지그룹에 기록한다(S138). 기록하는 방법은 블록맵페이지그룹에 속한 모든 블록에 지우기 연산을 수행하고 가장 하위주소의 블록맵페이지에 칩t의 블록맵정보를 기록하면 된다.

<84> 예비영역포맷이 완료되면(S100), 블록맵정보를 주메모리에 상주시킨다(S200). 처음 플래시 메모리를 사용하는 경우가 아니라면 각 칩의 블록맵정보는 가장 최근에 기록된 각 칩의 블록맵페이지와 같은 정보를 갖도록 각 칩에서 최신의 블록맵페이지를 검색하여 상주시킨다(S200).

<85> 블록맵정보를 주메모리에 상주시키는 방법은 도 17, 도 18, 도 19를 참조하여 설명한다. 먼저 플래시 칩 번호 t를 0으로 초기화시킨다(S210). 그리고, 칩t의 최신 블록맵정보를 주메모리에 상주시킨 후(S230), 해당 칩이 마지막 칩인지를 검사(S270)하여 마지막 칩이 아니라면, 플래시 칩 번호 t를 1 증가시키는 방식으로 마지막 칩에 대한 블록맵정보를 주메모리에 상주시킬 때까지 상기 과정을 반복한다.

<86> 칩t의 블록맵정보를 주메모리에 상주시키는 과정(S230)을 살펴보면 먼저 칩t의 블록맵정보의 카운트필드에 0을 기록하여 카운트 필드를 초기화한다(S232). 그리고 칩t의 블록맵페이지그룹으로부터 첫 번째 블록맵페이지를 읽는다(S234). 읽은 블록맵페이지의 유효성을 검사하여(S238) 유효한 블록맵페이지라면 칩t의 블록맵정보의 카운트와 읽은 블록맵페이지의 카운트를 비교한다(S240). 칩t의 블록맵정보의 카운트가 작다면 읽은 블록맵페이지의 정보가 칩t의 블록맵정보보다 더 최근의 정보라는 것을 의미하므로 블록맵페이지를 칩t의 블록맵정보에 복사하여 더 최근의 정보로 갱신한다(S242). 읽은 블록맵페이지가 마지막 블록맵페이지라면(S244) 칩t의 블록맵정보에 저장된 정보가 가장 최신의 블록맵페이지를 담고 있으므로 주메모리에 상

주시킨다. 읽은 블록맵페이지가 마지막 블록맵페이지가 아니라면 더 최신의 블록맵페이지가 존재하는지를 찾기 위하여 칩t의 다음 블록맵페이지를 읽는다(S236). 그리고 나서 상기 설명한 과정을 반복한다. 칩t의 유효 블록맵페이지인지를 검사하는 단계(S238)에서 유효페이지가 아니라면 이를 무시하고 칩t의 다음 블록웹페이지를 읽고(S236) 상기 설명한 과정을 반복한다. 읽기오류를 쓰기나 지우기 오류와 별도로 취급한다면, 읽기 오류를 처리하기 위한 프로세스(S246, S250, S260)가 추가된다. 유효블록맵페이지검사에서(S238) 읽기오류복구중 마크가 표시된 블록맵페이지라면(S246) 읽기오류를 복구하고(S250) 블록맵페이지에 읽기오류표시를 무효화한다(S260). 그리고 나서 다시 다음 블록맵페이지를 읽는다(S236). 한편, 유효 블록맵페이지도 아니고 읽기 오류도 아닌 경우에는 유효하지 않은 블록맵페이지이므로 다음 블록맵페이지를 읽는다. 다음 블록맵페이지를 읽고(S236) 다시 상기 S238에서 S244의 과정을 반복한다.

읽기 오류의 복구의 보다 상세한 과정은 도 19를 참조하여 설명한다. 먼저 칩t의 읽기오류블록을 지운다(S252). 블록맵정보를 주 메모리에 상주하는 과정에서 발견되는 읽기 오류는 이전의 읽기오류복구중을 표시한 단계이후에 복구작업 중 갑자기 전원이 차단되는 경우에 다시 전원이 들어올 때 발견된다. 읽기오류블록을 지운 후(S252) 칩t의 트랜스퍼블록을 읽기 오류블록으로 복사하고(s254), 칩t의 상기 찾아진 블록맵페이지에 무효 마크를 쓴다(S256). 그리고 난 후에 칩t의 보호영역을 읽기전용상태로 변경한다(S258).

<87> 블록맵정보가 주메모리에 상주하면(S200) 플래시 메모리의 보호영역을 읽기전용으로 설정한다(S280). 그리고, 플래시 연산을 할 수 있는 준비상태가 된다(S300). 도 3의 주메모리(400)의 플래시 어플리케이션(410)은 사용영역의 특정 주소를 플래시 연산을 하기 위해서는 플래시 장치관리자에 반물리주소를 넘겨주고 플래시 장치관리자는 해당 반물리주소를 바탕으로 물리주소를 생성한다(S400). 물리적 주소를 찾으면(S400) 플래시 연산을 한다(S500). 플래시 연산 중

에 오류블록이 발견되면 해당 반물리 주소를 오류블록이 아닌 예비블록을 사용할 수 있도록 매핑정보를 등록하고 오류를 수정하는 작업을 한다(S600). 오류블록을 처리하는 과정은 도 20 내지 도 22을 참조하면 다음과 같다. 만약 읽기 오류블록을 쓰기 또는 지우기 오류블록과 다르게 취급한다면, 칩t의 오류블록이 읽기 오류블록인지를 검사하고(S610), 읽기오류블록인 경우 칩t의 읽기 오류블록처리(S620)을 수행하고 그렇지 않은 경우에는 쓰기지우기오류블록처리(S660)를 한다. 읽기 오류블록을 쓰기 또는 지우기 오류블록과 동일하게 취급하는 경우에는 모든 오류블록에 대해서 칩t의 쓰기 지우기 오류블록 처리(S660)을 수행한다.

<88> 먼저 도 21을 참조하여 칩t의 읽기 오류블록처리(S620)를 살펴보면, 먼저 칩t의 보호영역을 읽기쓰기 상태로 변경한다(S622). 그리고 나서 칩t의 트랜스퍼블록을 지운(S624) 후, 칩t의 읽기 오류블록의 오류를 정정해서 칩t의 트랜스퍼블록에 복사한다(S626). 일정 비트 이하의 오류에 대해서는 에러정정코드로 오류를 정정할 수 있다. 이 때, 칩t의 미사용 블록맵페이지를 찾는데(S628), 만약 미사용 블록맵페이지가 없다면 가장 최신의 블록맵페이지가 포함된 블록이 아닌 블록에 지우기 연산을 한다. 그리고, 블록맵정보의 카운트필드에 1을 더한 후, 상기 지워진 블록의 최하위 주소의 미사용 블록맵페이지에 상기 칩t의 블록맵정보를 기록하고, 상기 블록맵페이지에 유효 마크를 한다. 그리고, 상기 블록맵페이지의 상위 블록맵페이지 중에서 미사용 블록맵페이지를 찾을 수 있다. 상기 찾아진 블록맵페이지의 트랜지션필드를 읽기오류복구중 상태로 블록맵필드1의 오류블록번호에 읽기오류블록번호를 블록맵필드1의 예비블록번호에 트랜스퍼블록번호를 기록하고(S630), 상기 찾아진 블록맵페이지에 유효 마크를 한다(S632). 그리고, 칩t의 읽기 오류블록을 지운 후(S634), 칩t의 트랜스퍼블록을 읽기 오류블록으로 복사하고, 칩t의 상기 찾아진 블록맵페이지에 무효 마크를 쓴다(S638). 그 때, (10) 칩t의 보호영역을 읽기전용상태로 변경한다(S640).

<89> 읽기오류복구중을 표시하는 S630단계는 S632단계이후에 복구작업 중 갑자기 전원이 차단되는 경우를 대비한 것이다. 마지막 무효마크를 하는 것은 이미 읽기오류블록은 이후에는 정상적으로 사용할 수 있기 때문에 읽기오류블록이 체크된 블록맵페이지를 무효마크하는 것이다. 도 18의 유효블록맵페이지검사에서(S238) 읽기오류복구중 마크가 표시된 블록맵페이지라면(S246) 상기 블록맵페이지의 블록맵필드1에 포함된 오류블록번호(166)에서 읽기오류블록번호를 얻어내어 상기 S632, 634, 636, 638의 단계를 거쳐 읽기오류를 복구한다. 읽기오류는 주로 보호영역에서 발생한 것에 대하여 설명하였으나, 만일 비보호영역에서 읽기오류가 발생한 경우에도 S622와 S640을 제외하고 나머지 단계를 동일하게 수행하여 읽기 오류블록을 처리할 수 있다.

<90> 다음으로 도 22을 참조하여 칩t의 쓰기 지우기 오류블록처리(S660)에서 먼저 칩t의 미사용 예비블록을 찾는다(S662). 미사용 예비블록은 칩t의 블록맵정보를 이용하여 찾을 수 있으며 그것은 다음과 같다.

<91> 오류블록이 칩t의 보호영역에서 발생했을 경우, 칩t의 블록맵정보에 등록되어 있는 블록맵필드중에서 이미 발생된 오류블록에 대하여 예비블록이 매핑되어 있는 오류블록번호(166)를 참조하여 미사용 예비블록을 찾는다. 바람직한 실시예에 있어서 이미 발생된 오류블록에 대하여 예비블록이 매핑되어 있는 경우에 가장 큰 번호를 갖는 예비블록부터 순차적으로 사용되지 않은 예비블록을 찾는다. 오류블록이 칩t의 비보호영역에서 발생했을 경우, 칩t의 블록맵정보에 등록되어 있는 블록맵필드중에서 이미 발생된 오류블록에 대하여 예비블록이 매핑되어 있는 오류블록번호(166)을 참조하여 미사용 예비블록을 찾는다. 바람직한 실시예에 있어서 이미 발생된 오류블록에 대하여 예비블록이 매핑되어 있는 경우에 가장 큰 번호를 갖는 예비블록부터 순차적으로 사용되지 않은 예비블록을 찾는다. 상기와 같은 방식으로 미사용 예비블록을 찾을 경

우에는 보호영역 및 비보호영역의 예비블록의 숫자를 미리 할당하지 않고 발생되는 오류블록에 맞도록 사용할 수 있는 장점이 있다.

<92> 미사용 예비블록을 찾으면(S662), 칩t의 블록맵정보에서 칩t의 미사용 블록맵필드를 찾는다(S664). 상기 S662 및 S664는 블록맵정보를 바탕으로 찾는다. 그리고 나서 블록맵정보를 갱신한다(S666). 블록맵정보의 갱신은 블록맵정보에 있는 찾아진 예비블록을 매핑하도록 예비블록과 오류블록의 번호를 적는다. 블록맵정보를 갱신한 후에 블록맵페이지그룹에서 사용할 수 있는 블록맵페이지를 찾고(S668), 오류블록이 보호영역에 포함되어 있는지 검사한 후(S670), 오류블록이 보호영역에 포함되어 있다면 칩t의 예비영역의 보호영역을 읽기 쓰기 상태로 변경한다(S672). 그러나 찾아진 미사용 예비블록이 읽기 쓰기가 가능한 비보호영역의 예비블록일 경우에는 상기 S672단계는 불필요하다. 여기서 미사용 블록맵페이지란 가장 최신의 정보를 담고 있는 블록맵페이지를 제외한 블록맵페이지들 중의 어느 한 블록맵페이지를 의미한다. 그리고, 칩t의 오류블록을 미사용 칩t의 예비블록으로 복사한다(S674). 그리고 나서 칩 t의 미사용 블록맵페이지에 칩t의 블록맵정보를 쓰고(S676), 칩 t의 미사용 블록맵페이지에 유효마크를 쓴다(S678). 마지막으로, 칩t의 예비영역의 보호영역을 읽기 전용 상태로 변경한다(S680). 오류블록을 예비블록으로 대체한 플래시 메모리는 오류블록이 존재하지 않는 것처럼 사용할 수 있다. 상기 (S668)단계에 쓰이는 블록맵페이지를 찾는 방법은 블록맵페이지의 트랜지션 필드를 사용하여 유효여부를 판단하고 유효한 것 중에서 카운트필드의 숫자가 가장 큰 것으로 판단한다. (S676)단계에서 빈 블록맵페이지에 블록맵정보를 기재하는데 이 때 기존에 저장된 블록맵페이지 중에서 가장 큰 카운트를 갖는 것보다 1

만큼 큰 카운트를 카운트필드에 적는다. 만일 비어있는 블록맵페이지가 없다면, 다른 블록맵페이지를 찾아서 쓰는데 거기도 꽉 찬 상태라면 최신의 블록맵페이지가 있는 블록이 아닌 블록에 대하여 지우기 연산을 하고 기록하면 된다.

<93> 예를 들어, 10개의 블록맵페이지를 갖고 있는 A블록과 B블록이 있다고 하자. 블록맵페이지를 사용하는 순서는 최하위주소부터 최상위주소로 사용한다고 할 때, A블록의 블록맵페이지를 모두 사용한 경우에는 B블록의 블록맵페이지를 사용한다. 이 때 각각의 블록맵페이지는 생성당시의 매핑정보를 가지고 있으므로 A블록과 B블록은 일종의 매핑정보 히스토리 블록역할을 한다. 따라서, A블록을 모두 사용하고 B블록의 마지막 블록맵페이지까지 사용한 경우에 새로운 오류블록이 발생하면 A블록을 지우고 거기에 새로운 매핑정보를 포함하는 블록맵페이지를 만든다. 이 때 만일 시스템의 오작동이나 전원이 차단되더라도 최소한 현재의 매핑정보를 제외한 가장 최근의 매핑정복 B블록에 존재하므로 플래시 메모리를 별 문제 없이 사용할 수 있다. 차후 현재 오류블록에 대해 플래시 연산을 하려고 하면 그 때 오류블록에 대하여 새롭게 예비블록을 매핑하고 그 매핑정보를 포함하여 블록맵페이지를 만들면 된다.

<94> 한편, 블록맵페이지그룹이 하나의 블록으로 되어 있다면 비어있는 블록맵페이지가 없을 때 블록맵페이지그룹을 지우고 다시 처음부터 쓸 수 있으나 만일 블록맵페이지를 지울 때 시스템의 오동작이나 전원이 꺼질 경우 기존의 매핑정보는 사라진다. 따라서 두 개 이상의 블록을 갖는 것이 바람직하며 두 개의 블록을 갖는 것이 가장 바람직하다.

<95> 본 발명이 속하는 기술분야의 당업자는 본 발명이 그 기술적 사상이나 필수적인 특징을 변경하지 않고서 다른 구체적인 형태로 실시될 수 있다는 것을 이해할 수 있을 것이다. 예를 들어, 이상의 실시예에서는 각각의 칩이 모두 예비영역을 갖는 것을 중심으로 설명하였으나 본 발명은 이에 한정되지 않고 예비영역을 특정한 칩에 몰아서 사용하는 경우에도 적용될 수 있다.

<96> 그러므로 이상에서 기술한 실시예들은 모든 면에서 예시적인 것이며 한정적이 아닌 것으로 이해해야만 한다. 본 발명의 범위는 상기 상세한 설명보다는 후술하는 특허청구범위에 의하여 나타내어지며, 특허청구범위의 의미 및 범위 그리고 그 등가개념으로부터 도출되는 모든 변경 또는 변형된 형태가 본 발명의 범위에 포함되는 것으로 해석되어야 한다.

【발명의 효과】

<97> 상기한 바와 같이 이루어진 본 발명에 따르면, 플래시 메모리의 사용 중에 오류블록이 생겼을 경우에 적은 플래시 메모리 연산으로 오류블록을 관리할 수 있어 시스템의 성능을 종전의 발명과 달리 향상 시킬 수 있다.

<98> 또한 읽기전용으로 설정된 영역(보호영역)이 있는 플래시 메모리의 보호영역에서 발생된 오류블록을 대체하여 할당되는 새로운 영역 또한 읽기전용으로 설정 할 수 있어 변경할 필요가 없는 중요한 데이터를 안전하게 관리할 수 있는 방법을 제공한다.

【특허청구범위】

【청구항 1】

(a) 플래시 메모리에 복수의 사용블록을 갖는 사용영역과 복수의 예비블록을 갖는 예비영역을 할당하고, 상기 사용영역 또는 상기 예비영역에서 발생하는 오류블록과 이를 대체하여 사용하는 예비블록과의 매핑정보를 저장하는 블록맵페이지를 복수개 포함하는 블록맵페이지그룹을 제공하는 단계;

(b) 상기 블록맵페이지그룹 중에서 소정의 규칙에 의해 선택된 블록맵페이지의 매핑정보를 메모리에 상주시키는 단계; 및

(c) 플래시 연산 중에 발생된 오류블록을 상기 상주된 매핑정보를 통해 찾은 미사용 예비블록과 매핑하고, 상기 상주된 매핑정보를 갱신하고 상기 갱신된 매핑정보를 상기 블록맵페이지그룹에 속하는 블록맵페이지에 기록하는 단계를 포함하는 플래시 메모리의 오류블록 관리방법

【청구항 2】

제1항에 있어서, 상기 (a)단계는 플래시 메모리에 복수의 사용블록을 갖는 사용영역과 복수의 예비블록을 갖는 예비영역을 할당하는 (a1) 단계;

상기 생성된 예비블록과 발생될 오류블록의 매핑정보를 기록할 블록맵필드를 예비블록마다 생성하고, 상기 생성된 블록맵필드들, 매핑정보의 시점을 판단하기 위한 카운트 필드, 및 유효성을 판단을 위한 트랜지션 필드를 포함하는 블록맵정보를 초기화시키는 (a2) 단계;

플래시 메모리에 존재하는 오류블록을 검사하여 발견된 오류블록들 및 상기 발견된 오류블록들을 대체하는 예비블록들과의 매핑정보를 상기 블록맵필드들에 각각 기록하고 상기 카운트 필드에 초기 카운트를 하여 블록맵정보를 생성하는 (a3) 단계;

상기 블록맵정보를 상기 블록맵페이지그룹의 어느 한 블록맵페이지에 기록하는 (a4) 단계를 포함하는 플래시 메모리의 오류블록 관리 방법

【청구항 3】

제1항에 있어서, 상기 블록맵페이지그룹은 최소한 두 개의 예비블록으로 구성되는 것을 특징으로 하는 플래시 메모리의 오류블록 관리 방법

【청구항 4】

제1항 내지 제3항 중 어느 한 항에 있어서; 상기 플래시 메모리는 복수의 칩으로 이루어고, 각 단계들은 각각의 칩마다 수행되는 것을 특징으로 하는 플래시 메모리의 오류블록 관리 방법

【청구항 5】

제1항에 있어서, 상기 (a) 단계에서 상기 사용블록들 및 상기 예비블록들 중 적어도 일부를 보호영역으로 지정하여 검증되지 않은 소프트웨어의 접근을 차단하는 것을 특징으로 하는 플래시 메모리의 오류블록 관리 방법

【청구항 6】

제1항에 있어서, 상기 (b) 단계는 블록맵페이지그룹 안의 블록맵페이지들 중에서 유효하면서 가장 최근에 기록된 블록맵페이지를 검색하여 검색된 블록맵페이지의 매핑정보를 메모리에 상주 시키는 것을 특징으로 하는 플래시 메모리의 오류블록 관리 방법

【청구항 7】

제6항에 있어서, 상기 유효하면서 가장 최근에 기록된 블록맵페이지를 검색할 때 각각의 블록맵페이지들의 트랜지션 필드에 기록된 값 및 카운트 필드에 기록된 숫자의 크기를 기준으로 검색하는 것을 특징으로 하는 플래시 메모리의 오류블록 관리 방법

【청구항 8】

제6항에 있어서, 상기 블록맵페이지그룹 안의 블록맵페이지들 중에서 유효하면서 가장 최근의 블록맵페이지를 검색하는 과정에서 유효하지 않으면서 읽기오류가 표시된 블록맵페이지를 발견할 경우, 상기 발견된 읽기 오류블록을 소정의 방법으로 복구하는 단계를 더 포함하는 것을 특징으로 하는 플래시 메모리의 오류블록 관리 방법

【청구항 9】

제6항 내지 제8항 중 어느 한 항에 있어서, 플래시 메모리가 복수의 칩으로 이루어진 경우에 각 단계들은 칩마다 수행되는 것을 특징으로 하는 플래시 메모리의 오류블록 관리 방법

【청구항 10】

제1항에 있어서, 상기 (c) 단계는 상기 상주된 매핑정보를 이용하여 미사용 예비블록을 찾는 (c1) 단계;

상기 (c1) 단계를 통해 찾아진 미사용 예비블록과 오류블록의 번호를 매핑하여 상기 상주된 매핑정보를 갱신하는 (c2) 단계;

상기 블록맵페이지 그룹의 미사용 블록맵페이지를 찾는 (c3) 단계;

상기 발생된 오류블록에 기록된 정보를 상기 찾아진 미사용 예비블록에 복사하는 (c4) 단계;

상기 갱신된 매핑정보를 포함한 정보를 상기 찾아진 미사용 블록맵페이지에 기록하는 (c5) 단계; 및

상기 미사용 블록맵페이지가 유효함을 표시하는 (c6) 단계를 포함하는 것을 특징으로 하는 플래시 메모리의 오류블록 관리 방법

【청구항 11】

제10항에 있어서, 상기 오류블록이 보호영역에서 발생된 경우에 상기 보호영역을 읽기 쓰기 가능 상태로 변경하고, 상기 (c6) 단계가 끝난 후에 상기 보호 영역을 다시 읽기 전용 상태로 변경하는 단계를 더 포함하는 것을 특징으로 하는 플래시 메모리의 오류블록 관리 방법

【청구항 12】

제10항 또는 제11항에 있어서, 플래시 메모리가 복수의 칩으로 이루어진 경우에 각 단계들은 칩마다 수행되는 것을 특징으로 하는 플래시 메모리의 오류블록 관리 방법

【청구항 13】

제1항에 있어서, 상기 (c) 단계는 발생된 오류블록이 읽기 오류블록인지 여부를 판단하는 (c1) 단계;

읽기 오류블록을 복구하기 위해 임시로 사용하는 예비블록을 지우는 (c2) 단계;

읽기 오류블록의 오류를 정정하여 상기 예비블록에 복사하는 (c3) 단계;

상기 블록맵페이지그룹 중에 미사용 블록맵페이지를 찾는 (c4) 단계;

상기 찾아진 블록맵페이지에 상기 읽기 오류블록과 상기 예비블록의 매핑정보 및 읽기오류 복구 상태임을 표시하는 (c5) 단계;

상기 읽기 오류블록을 지우는 (c6) 단계;

상기 예비블록을 상기 지워진 읽기 오류블록에 복사하는 (c7) 단계; 및

상기 찾아진 블록맵페이지에 무효임을 표시하는 (c8) 단계를 포함하는 것을 특징으로 하는 플래시 메모리의 오류블록 관리 방법

【청구항 14】

제13항에 있어서, 상기 읽기 오류블록이 보호영역에서 발생된 경우에 보호영역을 읽기 쓰기 가능 상태로 변경하고, 상기 (c8) 단계가 끝난 후에 상기 보호 영역을 다시 읽기 전용 상태로 변경하는 것을 특징으로 하는 플래시 메모리의 오류블록 관리 방법

【청구항 15】

최소한 하나 이상의 플래시 메모리 칩으로 이루어진 플래시 메모리의 오류블록을 관리장치에 있어서,

복수의 사용블록을 갖는 사용영역과 사용 중 발생하는 오류블록을 대체하기 위한 복수의 예비블록을 갖는 예비영역을 가지며, 사용 중 발생되는 오류블록과 이를 대체하는 예비블록과의 매핑정보를 저장하는 블록맵페이지를 복수 개 갖고 있는 블록맵페이지그룹을 상기 예비영역 안에 갖고 있는 플래시 메모리;

상기 블록맵페이지그룹 안에 존재하는 최근의 유효한 블록맵페이지의 매핑정보와, 오류블록이 발생될 때 이를 처리하기 위한 프로세스를 정의하는 플래시 장치관리자를 로딩하기 위한 메모리; 및

상기 플래시 메모리와 상기 메모리에 전기적으로 연결되어 있고 상기 플래시 장치관리자의 코드를 읽어 상기 플래시 메모리에 대한 플래시 연산 및 오류블록 관리를 하는 중앙처리장치를 포함하는 플래시 메모리의 오류블록 관리 장치

【청구항 16】

제15항에 있어서, 상기 블록맵페이지그룹은 최소한 2개 이상의 예비블록으로 구성된 것을 특징으로 하는 플래시 메모리의 오류블록 관리 장치

【청구항 17】

제15항에 있어서, 상기 중앙처리장치와 상기 플래시 메모리 사이에 전기적으로 연결되어 있으며 상기 중앙처리장치가 플래시 연산을 하는 동안 플래시 메모리에 다음 연산을 위한 데이터를 저장하고 있는 버퍼를 두 개 이상 가지고 있는 플래시 메모리 콘트롤러를 더 포함하는 것을 특징으로 하는 플래시 메모리의 오류블록 관리 장치

【청구항 18】

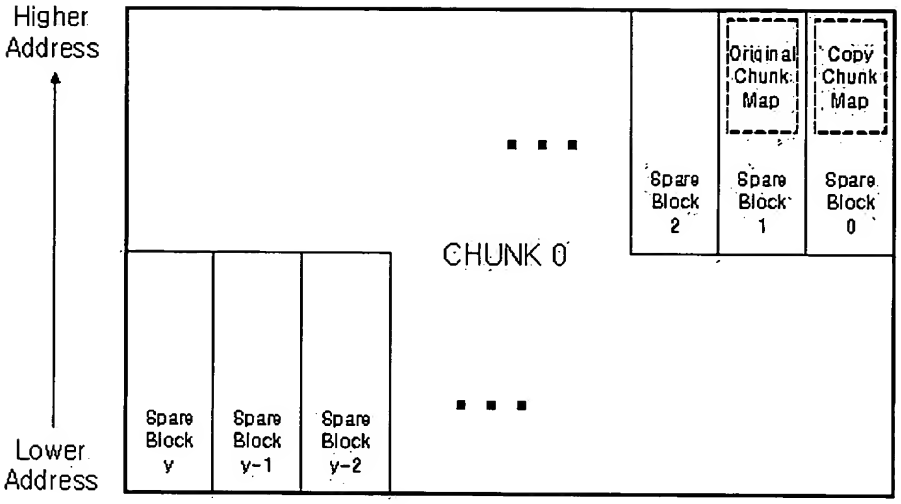
제17항에 있어서, 상기 플래시 메모리 콘트롤러는 상기 플래시 메모리의 일정영역을 보호영역으로 설정하여 상기 플래시 장치관리자에 의해 검증된 플래시 어플리케이션에 대해서만 상기 보호영역에 대한 플래시 연산을 허용하고 검증되지 않은 플래시 어플리케이션에 대해서는 상기 보호영역에 대한 플래시 연산을 허용하지 않는 것을 특징으로 하는 플래시 메모리의 오류블록 관리 장치

【청구항 19】

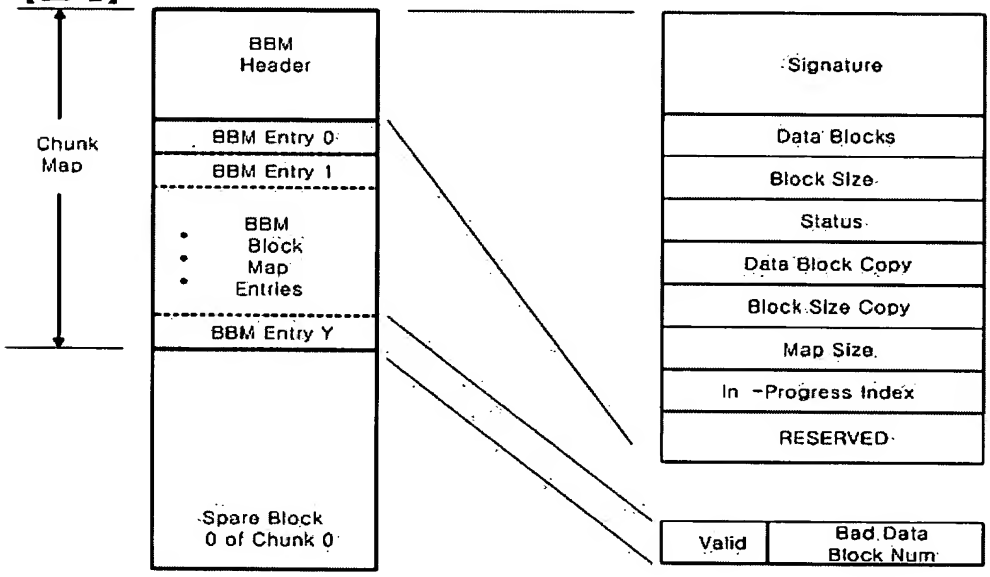
제15항에 있어서, 상기 플래시 메모리가 복수의 플래시 메모리 칩으로 이루어진 경우에 상기 플래시 메모리의 사용영역, 예비영역과 상기 예비영역 안의 블록맵페이지그룹은 각 플래시 메모리 칩마다 존재하는 것을 특징으로 하는 플래시 메모리의 오류블록 관리 장치

【도면】

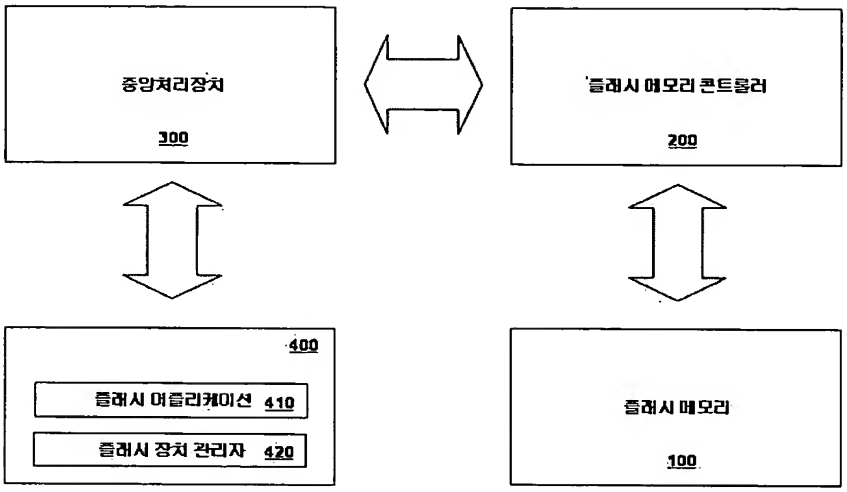
【도 1】


Higher Address
Lower Address
CHUNK 0
Spare Block y
Spare Block y-1
Spare Block y-2
Spare Block 2
Spare Block 1
Spare Block 0
Original Chunk Map
Copy Chunk Map

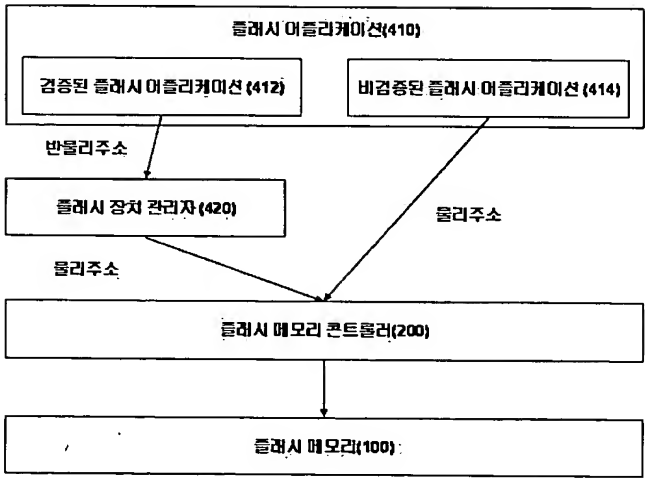
【도 2】


Chunk Map
BBM Header
BBM Entry 0
BBM Entry 1
BBM Block Map Entries
BBM Entry Y
Spare Block 0 of Chunk 0
Signature
Data Blocks
Block Size
Status
Data Block Copy
Block Size Copy
Map Size
In -Progress Index
RESERVED
Valid
Bad Data Block Num

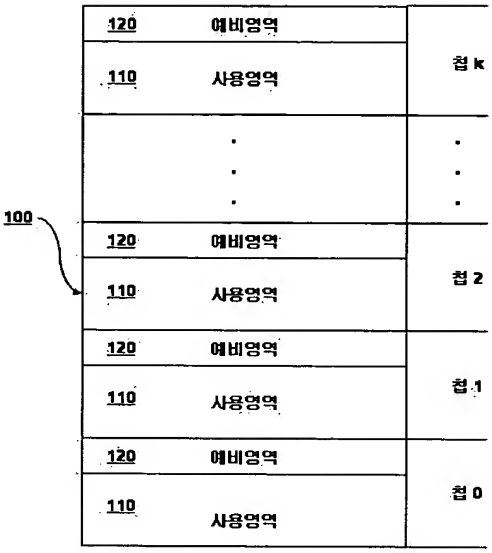
【도 3】


중앙처리장치
300
플래시 메모리 콘트롤러
200
400
플래시 어플리케이션 410
플래시 장치 관리자 420
플래시 메모리
100

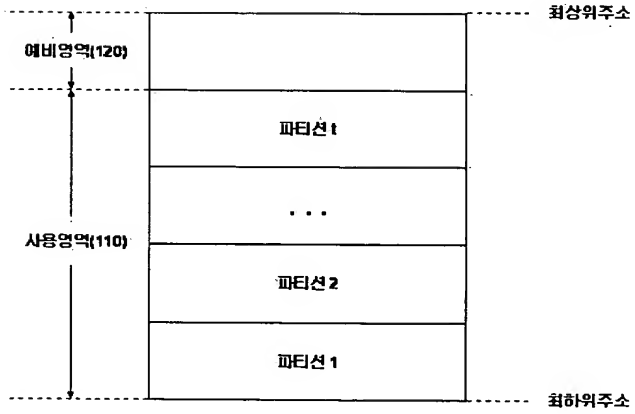
【도 4】


플래시 어플리케이션(410)
검증된 플래시 어플리케이션 (412)
비검증된 플래시 어플리케이션 (414)
반물리주소
플래시 장치 관리자 (420)
물리주소
물리주소
플래시 메모리 콘트롤러(200)
플래시 메모리(100)

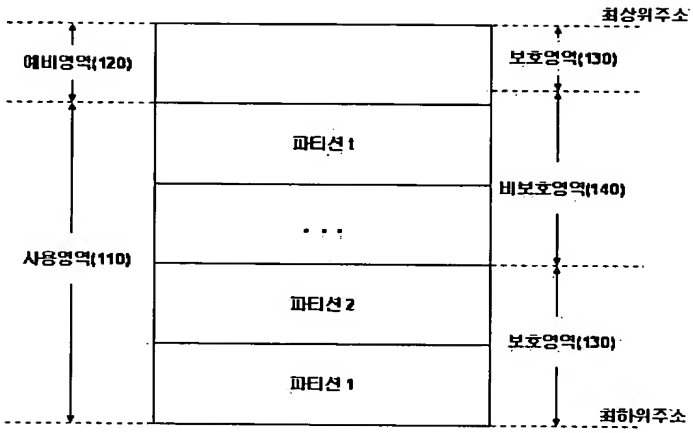
【도 5】


100
120
예비영역
110
사용영역
칩 k
칩 2
칩 1
칩 0

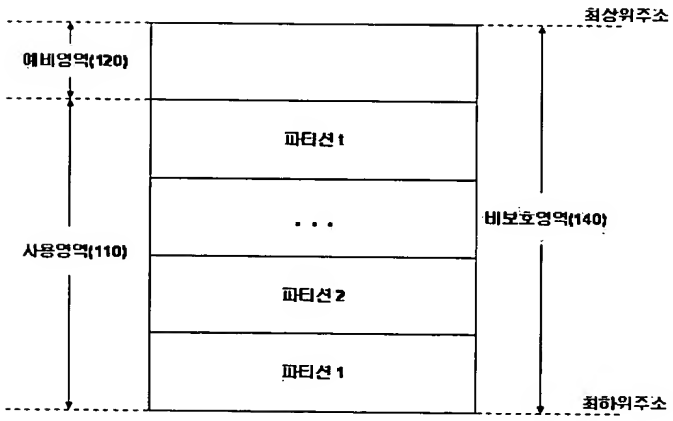
【도 6】


최상위주소
예비영역(120)
파티션 t
사용영역(110)
파티션 2
파티션 1
최하위주소

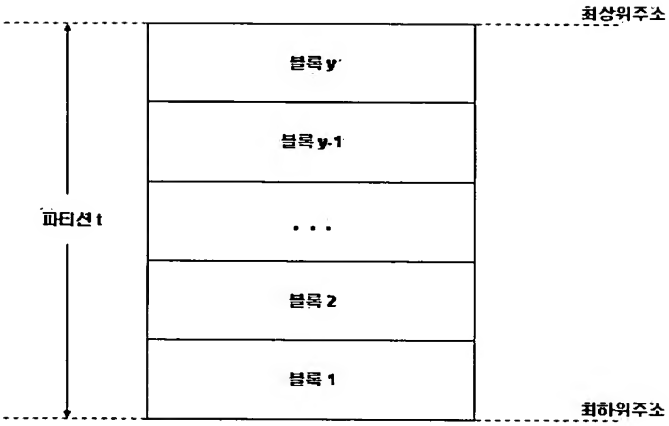
【도 7】


최상위주소
예비영역(120)
보호영역(130)
파티션 1
비보호영역(140)
. . .
사용영역(110)
파티션 2
보호영역(130)
파티션 1
최하위주소

【도 8】


최상위주소
예비영역(120)
파티션 1
. . .
비보호영역(140)
사용영역(110)
파티션 2
파티션 1
최하위주소

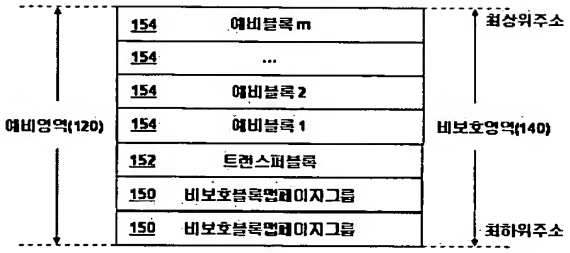
【도 9】


최상위주소
블록 y
블록 y-1
. . .
블록 2
블록 1
최하위주소
파티션 t

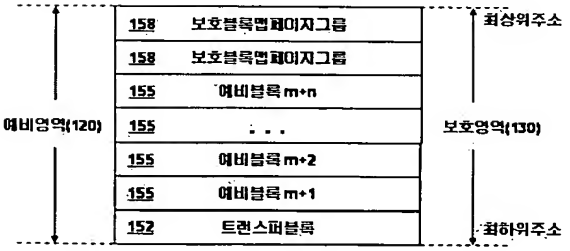
【도 10】


158 보호블록맵페이지그룹
158 보호블록맵페이지그룹
156 파티션정보블록
155 예비블록 m+n
155 ...
155 예비블록 m+1
154 예비블록 m
154 ...
154 예비블록 2
154 예비블록 1
152 트랜스퍼블록
150 비보호블록맵페이지그룹
150 비보호블록맵페이지그룹
최상위주소
보호영역(130)
비보호영역(140)
최하위주소
예비영역(120)

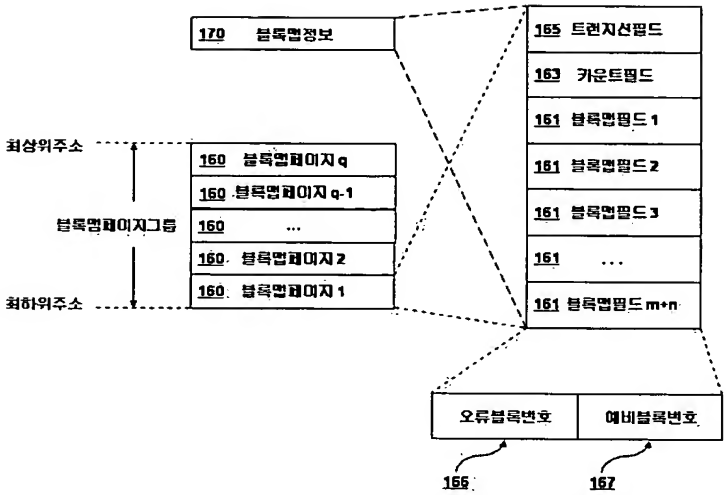
【도 11】


154 예비블록m
154 ...
154 예비블록2
154 예비블록1
152 트랜스퍼블록
150 비보호블록맵페이지그룹
150 비보호블록맵페이지그룹
최상위주소
최하위주소
예비영역(120)
비보호영역(140)

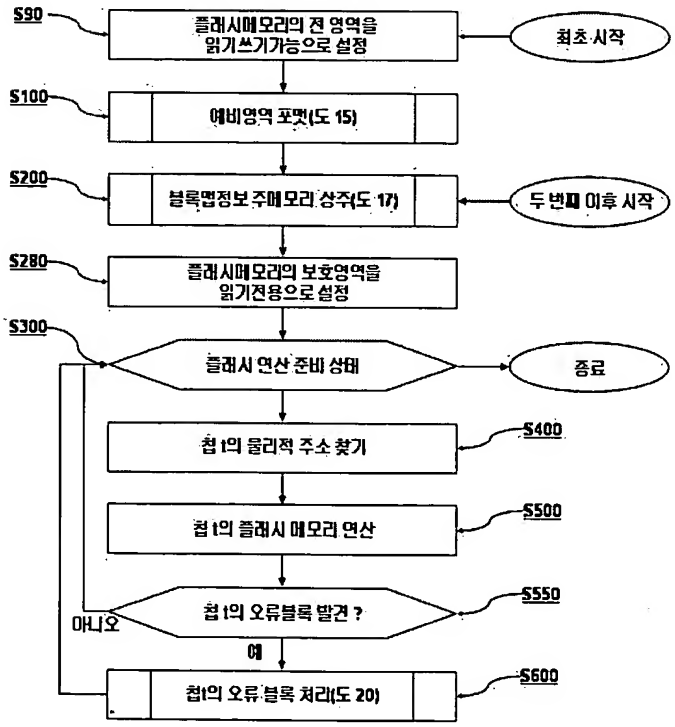
【도 12】


158 보호블록맵페이지그룹
158 보호블록맵페이지그룹
155 예비블록m+n
155 . . .
155 예비블록m+2
155 예비블록m+1
152 트랜스퍼블록
최상위주소
최하위주소
예비영역(120)
보호영역(130)

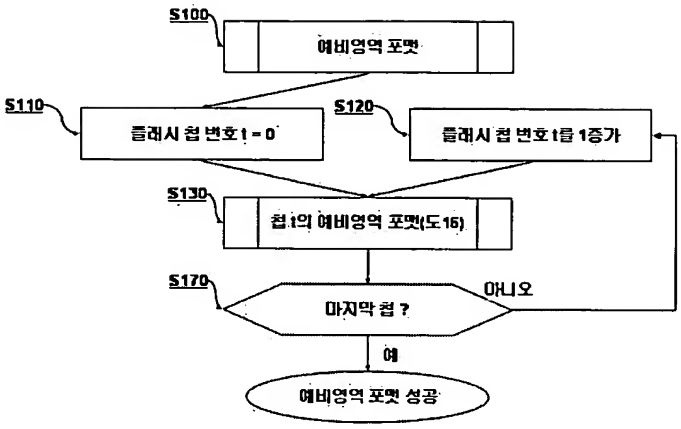
【도 13】


170 블록맵정보
165 트랜지션필드
163 카운트필드
161 블록맵필드 1
161 블록맵필드 2
161 블록맵필드 3
161 ...
161 블록맵필드 m+n
최상위주소
160 블록맵페이지 q
160 블록맵페이지 q-1
블록맵페이지그룹
160 ...
160 블록맵페이지 2
최하위주소
160 블록맵페이지 1
오류블록번호
예비블록번호
166
167

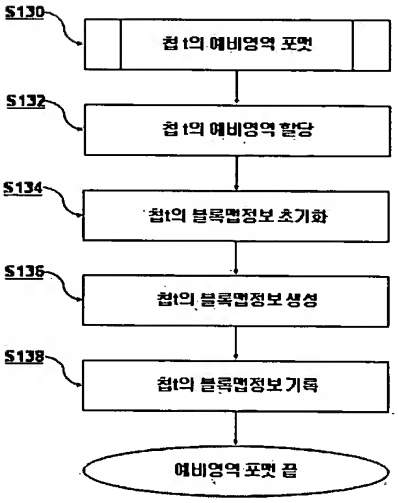
【도 14】


S90
플래시메모리의 전 영역을 읽기쓰기가능으로 설정
최초 시작
S100
예비영역 포맷(도 15)
S200
블록맵정보 주메모리 상주(도 17)
두번째 이후 시작
S280
플래시메모리의 보호영역을 읽기전용으로 설정
S300
플래시 연산 준비 상태
종료
칩 i의 물리적 주소 찾기
S400
칩 i의 플래시 메모리 연산
S500
칩 i의 오류블록 발견 ?
S550
아니오
예
칩 i의 오류 블록 처리(도 20)
S600

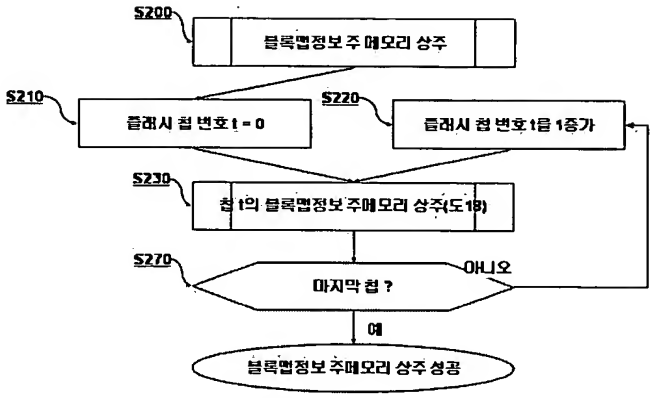
【도 15】


S100
예비영역 포맷
S110
플래시 칩 번호 i = 0
S120
플래시 칩 번호 i를 1증가
S130
칩 i의 예비영역 포맷(도16)
S170
마지막 칩 ?
아니오
예
예비영역 포맷 성공

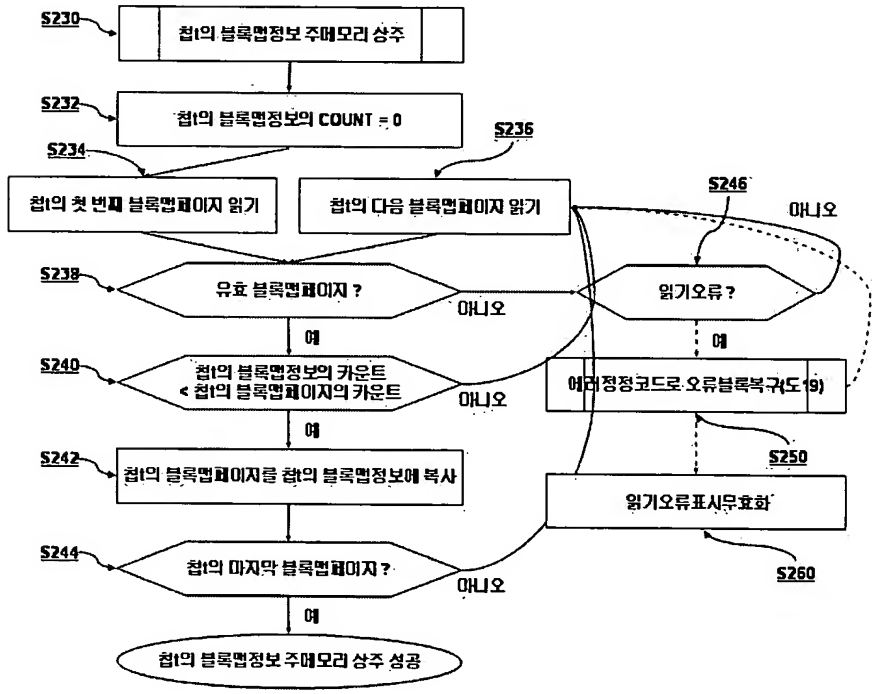
【도 16】


S130
칩 i의 예비영역 포맷
S132
칩 i의 예비영역 할당
S134
칩i의 블록맵정보 초기화
S136
칩i의 블록맵정보 생성
S138
칩i의 블록맵정보 기록
예비영역 포맷 끝

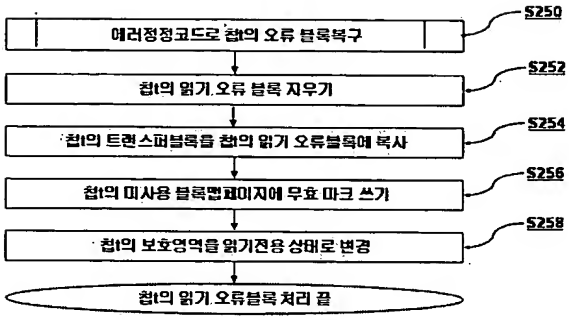
【도 17】


S200
블록맵정보 주메모리 상주
S210
플래시 칩 번호 t = 0
S220
플래시 칩 번호 t를 1증가
S230
칩 t의 블록맵정보 주메모리 상주(도 18)
S270
마지막 칩 ?
아니오
예
블록맵정보 주메모리 상주 성공

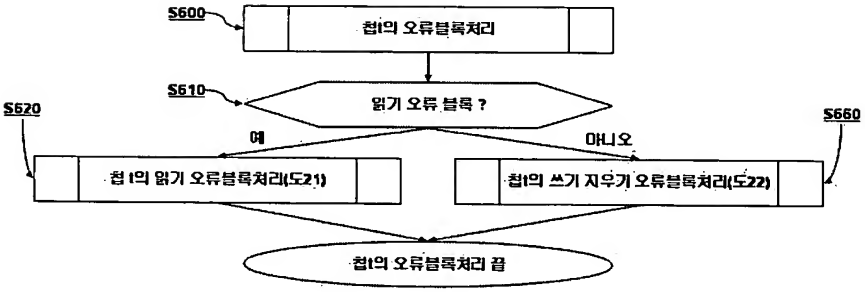
【도 18】


S230
칩 t의 블록맵정보 주메모리 상주
S232
칩 t의 블록맵정보의 COUNT = 0
S234
칩 t의 첫 번째 블록맵페이지 읽기
S236
칩 t의 다음 블록맵페이지 읽기
S238
유효 블록맵페이지 ?
아니오
예
S240
칩 t의 블록맵정보의 카운트 < 칩 t의 블록맵페이지의 카운트
아니오
예
S242
칩 t의 블록맵페이지를 칩 t의 블록맵정보에 복사
S244
칩 t의 마지막 블록맵페이지 ?
아니오
예
칩 t의 블록맵정보 주메모리 상주 성공
S246
읽기오류 ?
아니오
예
에러정정코드로 오류블록복구(도 19)
S250
읽기오류표시무효화
S260

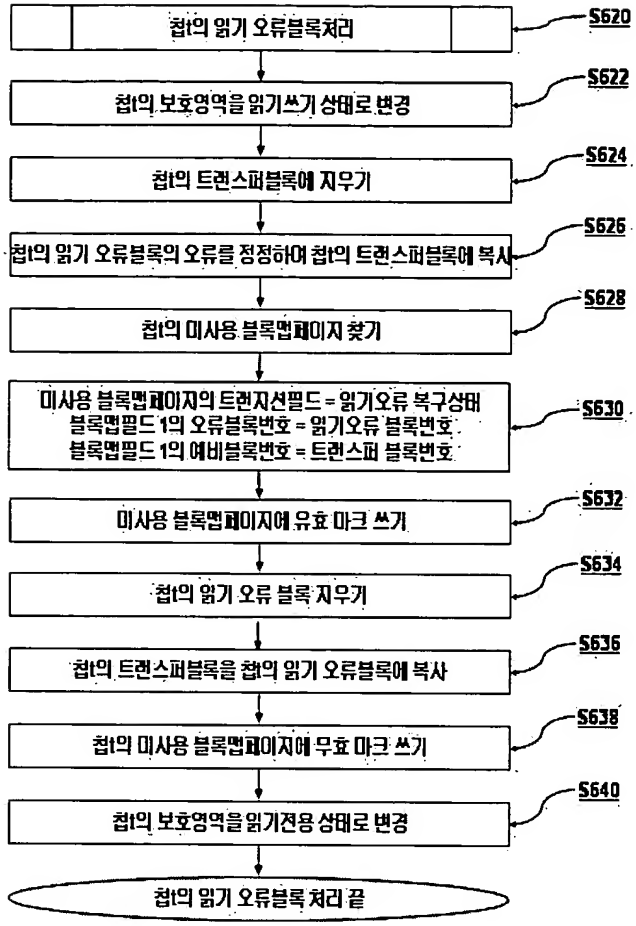
【도 19】


S250
에러정정코드로 챕i의 오류 블록복구
S252
챕i의 읽기 오류 블록 지우기
S254
챕i의 트랜스퍼블록을 챕i의 읽기 오류블록에 복사
S256
챕i의 미사용 블록맵페이지에 무효 마크 쓰기
S258
챕i의 보호영역을 읽기전용 상태로 변경
챕i의 읽기 오류블록 처리 끝

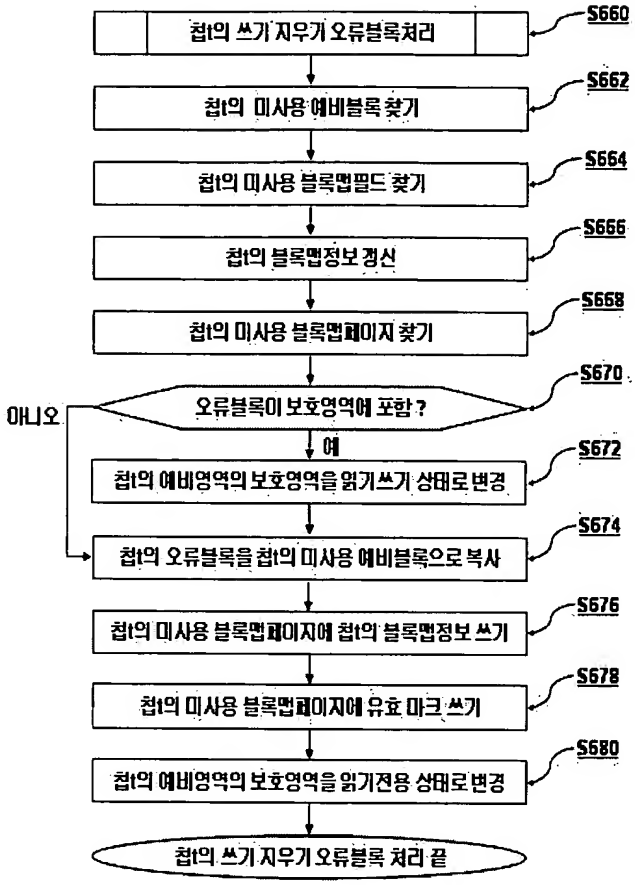
【도 20】


S600
챕i의 오류블록처리
S610
읽기 오류 블록 ?
S620
예
아니오
S660
챕i의 읽기 오류블록처리(도21)
챕i의 쓰기 지우기 오류블록처리(도22)
챕i의 오류블록처리 끝

【도 21】


첩t의 읽기 오류블록처리
S620
첩t의 보호영역을 읽기쓰기 상태로 변경
S622
첩t의 트랜스퍼블록에 지우기
S624
첩t의 읽기 오류블록의 오류를 정정하여 첩t의 트랜스퍼블록에 복사
S626
첩t의 미사용 블록맵페이지 찾기
S628
미사용 블록맵페이지의 트랜지션필드 = 읽기오류 복구상태
블록맵필드 1의 오류블록번호 = 읽기오류 블록번호
블록맵필드 1의 예비블록번호 = 트랜스퍼 블록번호
S630
미사용 블록맵페이지에 유효 마크 쓰기
S632
첩t의 읽기 오류 블록 지우기
S634
첩t의 트랜스퍼블록을 첩t의 읽기 오류블록에 복사
S636
첩t의 미사용 블록맵페이지에 무효 마크 쓰기
S638
첩t의 보호영역을 읽기전용 상태로 변경
S640
첩t의 읽기 오류블록 처리 끝

【도 22】


S660
첩t의 쓰기 지우기 오류블록처리
S662
첩t의 미사용 예비블록 찾기
S664
첩t의 미사용 블록맵필드 찾기
S666
첩t의 블록맵정보 갱신
S668
첩t의 미사용 블록맵페이지 찾기
S670
오류블록이 보호영역에 포함?
아니오
예
S672
첩t의 예비영역의 보호영역을 읽기쓰기 상태로 변경
S674
첩t의 오류블록을 첩t의 미사용 예비블록으로 복사
S676
첩t의 미사용 블록맵페이지에 첩t의 블록맵정보 쓰기
S678
첩t의 미사용 블록맵페이지에 유효 마크 쓰기
S680
첩t의 예비영역의 보호영역을 읽기전용 상태로 변경
첩t의 쓰기 지우기 오류블록 처리 끝